TCSL at the C@merata 2014 Task: A tokenizing and parsing framework to understand queries on sheet music

Nikhil Kini
Tata Consultancy Services Ltd.
Innovation Labs, Thane
nikhil.kini@tcs.com

ABSTRACT

We describe a system to address the MediaEval 2014 C@merata task of natural language queries on classical music scores. Our system first tokenizes the question to tag the musically relevant features in the question using pattern matching. In this stage suitable word replacements are made in the question based on a list of synonyms. Using the tokenized sentence we infer the question type using a set of handwritten rules. We then search the input music score based on the question type to find the musical features requested. MIT's music21 library [2] is used for indexing, accessing and traversing the score.

1. INTRODUCTION

When studying musicological analyses of works of western classical art music, there are frequent references to relevant passages in the printed score. Musicologists can refer to very complex aspects of a score within a text, and other experts know what passages they are talking about because they can interpret musical terminology and can then look through scores to find the passages in question. However, this can be time consuming. The long-term aim here is to develop tools that could facilitate the work of musicologists, by automatically identifying the passages that are referred to.

The problem may be defined as follows: given a computer representation of music as a score in a particular format (in our case, musicXML), and given a short English noun phrase referring to musical features in the score, search and list the location of all occurrences of the said musical features in the score. A formal, complete description of the task can be found at [1].

While a large body of work is available in other domains for Natural language understanding, as well as for searching through sheet music scores, we did not come across any work that combines these aspects. A survey of natural language understanding systems may be found in [3], and work done in non-trivial search on sheet music scores can be found in [4], [5], [6] and [7].

2. SYSTEM FRAMEWORK

Figure 1 presents the main modules of our system. Since we treat the problem as one of natural language understanding (of the question) and searching (through the musicXML), we define a set of question classes based on the searchable musical features, and propose a specific search method for each type of question.

Conference'14, Month 10–2, 2014, Barcelona, Spain.

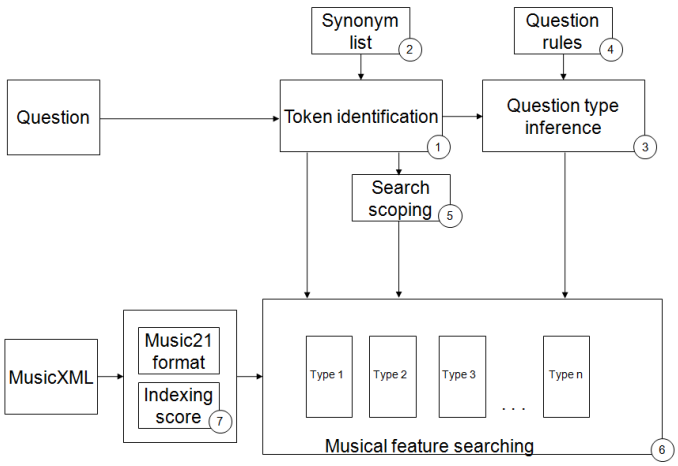

Figure 1. System for natural language sheet music querying

The main operations performed by our system are a) Identifying tokens in the question b) Inferring the question type c) searching through the sheet music and these are the main modules - Token identification (1), Question type inference (3) and Musical feature searching (6) respectively as shown in the Fig. 1. These are explained below along with the other operations - score indexing (7), search scoping (5) - and the input lists for synonyms (2) and question rules (4).

2.1 Identifying tokens in the question

In the tokenizing step (module (1) in Fig. 1), words representing musically important features are marked/tokenized. We use 3 or 4 letter markers for the token class. We specify the following classes:

Table 1. Token classes

Token class id	Class name	Matching Examples
PCH	pitch class	Bb, F natural
DUR	Duration	Semibreve, quarter
PRT	Part	Part I, Soprano I
RST	Rest	Rest
DIR	Direction	Ascending, descending
INT	Interval	Maj. 7th, octave
EXT	Extrema	Highest note, nadir

Token class id	Class name	Matching Examples
HRML	Harmony or melody	Harmonic, melodic
CLF	Clef	G-clef, Bass
SEQ, CSEC	Sequence	Then, before, consecutive
TRD	Triad	Triad
LYR	Lyrics	the lyrics "While love doth grant it", the word "cherish"
EXPR	Expressions	Fermata, Mordent
DYN	Dynamics	Forte, Pianissimo
CAD	Cadence	Perfect cadence, V-I cadence
TMSG	Time signature	3/4 time, common time
KYSG	Key signature	A change in key signature
CHNG	Change	A change in time signature
TXTR	Texture	polyphony, homophony
WIP	Work in progress[1]	

After tokenization, the sentence will contains tokens grouped with the value of the token, each token-value pair grouped by parentheses, and the token and the value will be separated by a comma. For e.g. "quarter note then half note then quarter note in the tenor voice" is output as "(DUR, quarter note) (SEQ, then) (DUR, half note) (SEQ, then) (DUR, quarter note) in the (PRT, tenor voice)". Another example is "melodic octave" becomes "(HRML, melodic) (INT, octave)".

2.2 Synonyms List

A list of synonyms is referred to during tokenizing for substituting words that refer to the same feature (module (2) in Fig. 1). This serves two purposes: 1) to cover all manners of asking for the same feature and 2) standardizing the different ways of asking for the same thing so that specifying the subsequent modules becomes simpler. The list of synonyms can be updated as new ways of asking the same feature are discovered when users actually query the system. When we come across a synonym, a chosen 'standard' word is substituted in its stead.

Table 2. Synonyms substitution list (non-exhaustive)

Synonyms	Substitution
Sharp	#
Flat	-
Asc, rising, rise, leap	ascending
Desc, falling, fall	descending
Right hand	G-Clef
Left hand	F-Clef
Quarter note	Quarter

[1] As a specification which will evolve, we give this placeholder to allow for future expansion

How do we choose this 'standard' substituted word? This will depend on the implementation. Since we are using music21 as our search implementation library, we have chosen words which are understood by music21. For example, for the note 'B flat', music21 understands this as 'B-'. Hence we substitute '-' for 'flat'. Also, right and left hand are assumed to mean G-Clef and F-Clef as per music21 defaults.

Taking the same example as before - "quarter note then half note then quarter note in the tenor voice" - with synonym substitution in the tokenization stage, the output is: "(DUR, quarter) (SEQ, then) (DUR, half) (SEQ, then) (DUR, quarter) in the (PRT, tenor)".

2.3 Inferring the question type

The tokenized output (with standard word substitution) is the input to the module which infers the question type (module (3) in Fig. 1). A handcrafted set of rules were used to guess what type of question is asked based on the constituent tokens (see section 2.4). Looking at all questions available to us so far - task description, training set, test set - we specify the following types of questions. New types may be added on discovery.

Table 3. Types of question

Question Type	Example questions
simple note (incl. pitch with or without duration, and also only duration including rests)	G-clef F natural, demisemiquaver B, dotted minim in the bass, sixteenth note, minim rest
note with expression	F# with a mordent, dotted B with an accent
interval (harmonic)	harmonic interval of a perfect fourth
interval (melodic)	falling third, rising second
lyrics	the lyrics "While love doth grant it"
extrema	apex in the tenor voice
time signature	a phrase in 3/4, change in time signature
key signature	a phrase in C Minor, change in the key signature
cadence	perfect cadence, imperfect cadence
triads	tonic triad, triad in first inversion
texture	counterpoint, melody with accompaniment
bar with dynamics	a passage in pianissimo, change from mezzoforte to pianissimo
consecutive notes	ascending G B D, B followed by D followed by G
combination of the above	F# followed two crotchets later by a G, a note repeated twice
work-in-progress[1]	

2.4 Question rules

Based on the tokens present in the question phrase, we can write rules to guess the type of the question. For simple questions made up of only one question type above (all except the last) this is straightforward.

Table 4. Rules for determining question type

Question Type	Question Rule
interval (harmonic)	One INT token, and optionally a (HRML, harmonic) token-value pair. Some tokens are optional because suitable defaults are defined
interval (melodic)	The token-value pair (HRML, melodic) and one INT token with an optional DIR token
lyrics	The token LYR
extrema	The token EXT
time / key signature	The TMSG / KYSG token with an optional CHNG token
cadence	The token CAD
triads	The token TRD
texture	The token TXTR
bar with dynamics	The token DYN
simple note	Contains tokens PCH, DUR, PCH and DUR, or RST and DUR, and no other tokens
consecutive notes	A simple note (as defined above) and another simple note separated by the SEQ token. Alternatively, the CSEC token followed by more than one simple note
note with expression	A simple note with a EXPR token
combination of the above	*work in progress*

For the phrases which contain a combination of elementary question types, some parsing capability might be necessary. We will address this in future work.

2.5 Search scope

An important part to getting the right answer is limiting the search scope. This is done by the module (5) in figure 1. For e.g. for the question "A sharp in the Treble clef" we are not just looking for any A#, but particularly in the Treble clef. Our tokens PRT and CLF can be used to scope the search. We look only in these parts during searching or we filter only those search results as answers which are within this search scope. Another possible scope restricting phrase is “on the word”, as in the test question “G on the word 'praise"”. This was not considered by us in the development of the system. It might be useful to introduce a SCP meta-token for scope, which can be used to scope the search when multiple question types are involved. For e.g., F# followed two crotchets later by a G.

2.6 Searching for the answer

Once the question type is inferred, the last step is searching the musicXML score for the identified token/token combination. This step is still a work in progress and we were unable to device a specification. Currently, it is implementation specific. We make extensive use of music21 capabilities. This step has a lot of room for optimization. As one of the optimizing steps, the baseline system creates a list of all notes in the score so that look up for simple notes becomes easy. Another possible optimization is a reverse look up of all or select musical features (the key being the musical feature and the values being all the passages where the feature occurs)

2.7 Score index

A score index is a list of all the notes in the score stored with the following associated information for each note:

note name, note letter, accidental, pitch class, note octave, bar, offset, note length, part number, part id and whether this is a rest or a note. (This terminology is as defined in music21).

3. APPROACH AND IMPLEMENTATION

The system given above was outlined after seeing the training data and base system provided by the organizers. A training set of 6 sheet music scores and a total of 36 questions were provided. The choice of music21 for implementation of our musicXML search, especially, is due to the base system implementation.

In many natural language understanding application, it is standard practice to run the input through a named entity recognition parser. While it was proposed that we could optionally use a parser such as the Stanford NER parser, we avoided it because a) the questions were noun phrases rather than entire sentences and as such, the services of the parser seemed like overkill b) Highlighting that a particular term is a musical feature would not be something an NER would do out of the box.

3.1 Preparing a list of possible questions and question types

We first prepared a list of possible questions from the sample questions in the task description and the training set. We observed that questions could be grouped into certain types based on the musical features that need to be queried and the relationship between them. The problem of understanding the intent of the query was reduced to identifying the type that the query belonged to. By observing examples of queries in this training set and also the examples in the Task Description, we formulated a set of question types. Once the classes were recognized, the intent was to design a specific search for each type of question, with as much reuse as possible.

Our implementation did not cover all search cases nor all classification cases. It is still a work in progress.

3.2 System at the training stage

Given below in Table 5 are sample outputs of the system at the training stage. The system correctly classified 27 out of 36 questions. Since no provision was made to identify "combinations of the above" type of questions, such questions were either not classified or wrongly classified. The consecutive notes question "a note repeated twice" was also not classified. The phrase "Simultaneous rhythm in five voices" was not understood by us, and we did not include it in the system for classification.

In Table 6 are given some examples outputs from our tokenization module and from the question identification module. At the time of the release of the test set, we could implement search for the following types of questions: simple note, interval (melodic), lyrics, extrema and consecutive notes. The other searches are yet unimplemented, and a work in progress.

Table 5. Training data - Type of questions

Class	Example questions	No. of Qs
interval (harmonic)	harmonic interval of a perfect fourth	4
simple note (pitch with/without duration, only duration)	G-clef F natural, demisemiquaver B, dotted minim in the bass, sixteenth note	4
interval (melodic)	three rising seconds followed by a falling third	9
lyrics	the lyrics "While love doth grant it"	1
extrema	apex in the tenor voice	6
cadence	perfect cadence, imperfect cadence	2
consecutive notes	ascending G B D, a note repeated twice	4
combination of the above	F# followed two crotchets later by a G, dotted crotchet tied with a crotchet	5
Unknown	simultaneous rhythm in five voices	1

Table 6. Training data - Sample tokenization and Question type identification outputs

Input question noun phrase	Output of tokenizer	Question type
harmonic interval of a perfect fourth	(HRML, harmonic) interval of a (INT, perfect fourth)	interval (harmonic)
F3 sharp in the "Alt"	(PCH, F3#) in the "(PRT, alt)"	simple note
Rising melodic octave	(DIR, ascending) (HRML, melodic) (INT, octave)	interval (melodic)
the lyrics "While love doth grant it"	the (LYR, "While love doth grant it")	lyrics
Nadir in the Soprano	(EXT, nadir) in the (PRT, soprano)	extrema
Dotted quaver E followed by semiquaver F sharp	(DUR, dotted quaver) (PCH, E) (SEQ, followed by) (DUR, semiquaver) (PCH, F#)	consecutive notes
Perfect cadence	(CAD, perfect)	cadence

The question phrases that were not classified are:

i. F# followed two crotchets later by a G
ii. ascending G B D
iii. dotted crotchet tied with a crotchet
iv. 3 parts in unison
v. simultaneous rhythm in five voices
vi. a note repeated twice
vii. three rising seconds followed by a falling third
viii. three ascending seconds followed by a fall of a third
ix. two or more consecutive half-steps in a voice

3.3 System after looking at the test set of questions

We saw that the test questions did not introduce any new type of question (with the exception of the "note with expression" subtype of the "simple note" type).

At the training stage, our system contained all of the tokens except the expression token "EXPR" which was added after looking at the test questions. Another change made to the system after looking at the test questions was adding the synonym substitution G-clef for "right hand" and F-Clef for "left hand". Apart from this we made no changes to the system after the release of the test set.

We have, however, noted points of improvement for the system. For example, with the current implementation, the system fails to recognize the phrase F3 sharp (while F#3 would get recognized). This is something that can be easily remedied by minimal changes to the implementation.

4. RESULTS AND DISCUSSIONS

Upon release of the results, we saw that the organizers had also used a scheme of classification for the questions. Reconciling the organizers and our question types, we see the following correspondence:

Table 7. TCSL and organizer question type mapping

Organizer question type	TCSL question type
simple_pitch	simple note
simple_length	
pitch_and_length	
perf_spec	simple note with expression
stave_spec	simple note with a staff scope
word_spec	simple note with a lyrics scope
followed_by	consecutive notes
melodic_interval	interval (melodic)
harmonic_interval	interval (harmonic)
cadence_spec	cadence
triad_spec	triad
texture_spec	texture

Table 8. Results for test set

	Beat			Measure		
Type	**Precision**	**Recall**	**F1 score**	**Precision**	**Recall**	**F1 score**
simple_length	0.979	0.988	0.983	0.991	1	0.995
simple_pitch	0.959	0.963	0.961	0.982	0.986	0.984
pitch_and_length	0.723	0.892	0.799	0.754	0.93	0.833
stave_spec	0.661	0.987	0.792	0.661	0.987	0.792
melodic_interval	0.894	0.683	0.774	0.904	0.691	0.783
followed_by	0.733	0.688	0.710	0.842	0.789	0.815
word_spec	0.261	1	0.414	0.261	1	0.414
perf_spec	0.0657	0.897	0.122	0.0657	0.897	0.122
harmonic_interval	0	0	N/A	0	0	N/A
cadence_spec	0	0	N/A	0	0	N/A
triad_spec	0	0	N/A	0	0	N/A
texture_spec	0	0	N/A	0	0	N/A
all	0.633	0.821	0.715	0.652	0.845	0.736

So we see that as far as the test questions go, we had all possibilities covered. There were also no questions of the "combination of the above" type.

Table 8 shows the beat and measure precision recall scores for the results produced by our system for the test set. Beat and measure correctness are defined as follows: A particular passage it can be considered bar-correct if it starts in the bar where the requested feature starts and ends in the bar where the requested feature ends. A passage is considered beat-correct if it starts at exactly the correct beat in the start bar and also ends at the correct beat in the end bar.

The strongest performance is seen in the 'simple notes' category (simple pitch, simple length, pitch and length). This is no surprise as these question phrases are the easiest to handle.

Perf_spec questions are simple note with expression type questions (involving for example, mordant and trill). Although these were not handled by our implementation, some results were returned because the system fell back to simple the simple note type, which explains the non-zero precision and recall. For e.g. “F trill” returns *all* F notes.

Word_spec is pitch/length occurring over a certain word in the lyrics. Our implementation treated this as a simple note and hence, although the recall is 1, precision is poor because all the notes of the particular pitch/duration are returned (and not just the one on the specified word)

Followed_by is equivalent to consecutive notes. Melodic_interval is a type with our system too, and the system performs decently on both these types.

Although search was not implemented for harmonic_interval, cadence_spec, triad_spec and texture_spec, nearly all of these question types were correctly classified by our system (exceptions are given below). No answers were returned for these types of questions, which results in the zero scores seen in the table.

Fewer questions were not classified in the test set than in the training set:

i. melody with accompaniment
ii. melody with accompaniment
iii. falling tone
iv. minim on the word "Der"
v. minim B on the word "im"
vi. eighth note on the word "che"
vii. word "Se" on an A flat
viii. G on the word "praise"

5. CONCLUSION AND RECOMMENDATIONS

The system implemented based on the specifications in this pa per performs decently on single musical feature retrieval. A study of the errors in this implementation might even be able to take the precision and recall for such simple types to 1, and this will be the aim of the next cycle of development.

While our system performs well on the simple question phrases, the more complex question phrases still need work. As a question grows more complicated to include multiple musical features, we will need to evolve a more complex parsing strategy to identify questions. It is possible that the specification of the system will need to be revisited to take into account all the possibilities.

The scope of the system specification is limited mainly to what we have observed in the task description and the training set, and these are in no way exhaustive of the types of queries that can be asked. However, the aim remains to specify a modular system that can accommodate changes which will naturally result from wider adoption and the resultant desire for a more powerful natural language querying system.

We are also desirous of extending the capabilities of the system to naturally learning new types of questions and new ways of asking the same type of questions.

6. ACKNOWLEDGMENTS

Many thanks to Dr. Sunil Kumar Kopparapu, my supervisor, for his help in shaping this paper.

7. REFERENCES

[1] Sutcliffe, R., Crawford, T., Fox C., Root, D. L., and Hovy, E. 2014. Shared Evaluation of Natural Language Queries against Classical Music Scores: A Full Description of the C@merata 2014 Task. In *Proceedings of the C@merata Task at MediaEval 2014*.

[2] Cuthbert, M. S., & Ariza, C. 2010. music21: A toolkit for computer-aided musicology and symbolic music data.

[3] Allam, A. M. N., & Haggag, M. H. (2012). The Question Answering Systems: A Survey. International Journal of Research and Reviews in Information Sciences (IJRRIS), 2(3).

[4] Gabriel, J. 2013. Large data sets & recommender systems: A feasible approach to learning music? in *Proceedings of the Sound and Music Computing Conference* 2013, SMC 2013, Logos Verlag Berlin, Number 2, Stockholm, Sweden, p.701–706 (2013)

[5] Downie, J. S. (1999). Evaluating a simple approach to music information retrieval: Conceiving melodic n-grams as text (Doctoral dissertation, The University of Western Ontario).

[6] Viro, V. (2011). Peachnote: Music Score Search and Analysis Platform. In ISMIR (pp. 359-362).

[7] Ganseman, J., Scheunders, P., & D'haes, W. (2008). Using XQuery on MusicXML Databases for Musicological Analysis. In ISMIR (pp. 433-438).