The C@merata task at MediaEval 2017: Natural Language Queries about Music, their JSON Representations, and Matching Passages in MusicXML Scores

Richard Sutcliffe[1], Donncha S. Ó Maidín[2], Eduard Hovy[3]
[1]University of Essex, UK
[2]University of Limerick, Ireland
[3]Carnegie-Mellon University, USA
rsutcl@essex.ac.uk, donncha.omaidin@ul.ie, hovy@cmu.edu

ABSTRACT

The C@merata task at MediaEval started in 2014 and is now in its fourth year. It is a combination of Natural Language Processing and Music Information Retrieval. The input is a short query ('six consecutive sixths in the right hand in bars 1-25') against a classical music score in MusicXML. The required output is a set of matching passages in the score. There are 200 queries and 20 scores each year. There were several innovations for 2017: First, some queries such as cadences required an answer which was a point in a score rather than a passage; second, queries were contributed by participants as well as by the organisers; third, some of the queries were directly taken from real texts such as articles and webpages; fourth, the organisers provided experimental representations of the input queries in the form of JSON feature structures. These capture many aspects of the queries in a form which is much closer to an MIR query. There were just two participants in the evaluation, and scores were understandably low given the considerable difficulty of the queries. However, this year we have significantly advanced our knowledge of how music is talked about in natural language texts, how these relate to MIR queries, and how to go about converting a text into a query.

1 INTRODUCTION

The C@merata evaluations are concerned with the relationship between Natural Language Processing (NLP) and Music Information Retrieval (MIR). Descriptions of classical music in books, papers, reviews and web pages are often very detailed and technical; however, experts can readily understand how they relate to the music. How can computers attain an equal level of understanding? The C@merata task aims to answer this question.

Each year, there are 200 questions against twenty classical music scores in MusicXML. Participants have to build a system which can return a set of one or more answer passages for each query. In previous years, each passage marked the start and end in the score of one answer to the question. This year, in addition, some answers are points in the score. For example, it is sometimes not clear where a cadence starts and ends; what is clear is the instant when the V-I transition occurs. Therefore, by using points, we can reduce ambiguity. The use of points allowed new types of query such as key changes, which are clearly points not passages.

In the next section we briefly describe the task, including the method of evaluation. After that, we outline the preparation of the Gold Standard data, comprising queries, scores and answers. Next, we briefly describe the work on feature structure representations for queries, expressed in JSON. The details of the 2017 campaign are then presented, together with the results. Finally, we draw conclusions from the 2017 C@merata task.

2 2017 TASK

The C@merata task has remained almost unchanged since 2014 and detailed descriptions can be found in previous papers [4,5,6,7,8,9]. We summarise the main points here. There are 200 questions, each one being a single noun phrase in English. Half of the questions use American terminology (quarter note, measure) while the other half use English terminology (crotchet, bar).

There are twenty MusicXML scores, and ten queries are set against each one. Participants must answer each query by means of one or more answer passages or answer points. There are now two forms of answer, passages and points. A passage specifies part of a score, beginning and ending at specific places. For example, [4/4,1,1:1-2:4] means we are in 4/4 time, divisions is set to one (i.e. we are measuring in crotchets) the passage starts before the first crotchet beat of bar one (1:1) and the passage ends after the fourth crotchet beat of bar two (2:4).

A point specifies the instant at which something happens in the score. For example, [3/4, 1, 7a3] means we are in 3/4 time, beating in crotchets, and the point falls in bar 7 after the third crotchet beat. In other words, this is the very end of bar seven. Of course, we have to consider whether that is the same as the start of bar eight, which we would write as [3/4, 1, 8b1]. There are subtle differences; a repeat mark could be considered as being at the end of a bar and not at the start of the next bar, while a key signature would be at the start of a bar, not at the end of the previous one.

For the 2017 task we resolve this ambiguity by stating that all points must be specified in the 'a' form (e.g. '7a3') except the very start of the piece, which by definition will need to be in the 'b' form.

We have reflected on the significance of points vs. passages in the current campaign. Firstly, clefs, key signatures and time signatures are all points, because they all have zero length.

MediaEval'17, September 2017, Dublin, Ireland

Changes of clef, key signature etc. in the middle of the piece are also points. Grace notes are points, because, while they do have a length in performance, they have zero length from the perspective of the beat arithmetic within the bar in which they occur. A particularly interesting example can be found towards the end of Der Dichter Spricht from *Kinderszenen* by Robert Schumann. This is an extended passage lasting perhaps sixteen seconds, all taking up no beats in the score! Cadences are points not passages, because it is always clear where the transition from the V chord to the I chord takes place as it lies immediately before the start of the I chord. On the other hand, the start of the V chord may not be that clear, leading to problems if a cadence is specified as a passage with a beginning and end. Interestingly, this problem seems not to have come to light in the context of music theory exams and oral tests; the examiner may ask for a cadence to be identified (as perfect or plagal for example) but they never ask exactly where it is, as this is considered to be obvious.

The next issue concerning points is that the start of a bar is not the same as the end of the previous one, as we have already mentioned. Finally, graphic symbols such as dynamics (p,f) could have a position which was the closest point in the score (in the case of printed scores).

Once a system has answered a question, we need a method of scoring the passage or passages returned. We use an automatic evaluation procedure. A passage is *beat correct* if it starts at the correct beat in the correct start bar and it ends at the correct beat in the correct end bar. So, if the correct answer is a crotchet, the passage must start immediately before the crotchet in question and it must end immediately after it. Similarly, a passage is *measure correct* if it starts in the correct start bar (but not necessarily at the correct beat) and ends in the correct end bar. The notion of 'beat correct' is used as the basis for the strict evaluation measures, while 'measure correct' is for the lenient evaluation measures. Based on beat correct and measure correct, we can define the following:

Beat Precision (BP) as the number of beat-correct passages returned by a system, in answer to a question, divided by the number of passages (correct or incorrect) returned.

Beat Recall (BR) is the number of beat-correct passages returned by a system divided by the total number of answer passages known to exist.

Beat F-Score (BF) is the harmonic mean of BP and BR.

Measure Precision (MP) is the number of measure-correct passages returned by a system divided by the number of passages (correct or incorrect) returned.

Measure Recall (MR) is the number of measure-correct passages returned by a system divided by the total number of answer passages known to exist.

Measure F-Score (MF) is the harmonic mean of MP and MR.

3 PREPARATION OF GOLD STANDARD

The Gold standard consists of twenty scores, ten questions against each, and a complete list of answer passages for each question. The scores chosen can be seen in Table 3. This year we decided to re-use some scores from earlier years, as indicated in the 'Origin' column. There were two from 2014, one from 2015, and seven from 2016, with the remaining ten being new this year. Because of our wish to set questions based on real documents, we were particularly looking for well-known scores so that there were plenty of suitable texts. For example, symphonies by Haydn, Mozart and Beethoven together with string quartets by the same composers seemed likely to fit our criteria. However, such scores are not that readily available in high-quality MusicXML scores. Hence, we wanted to re-use such scores wherever possible.

As we have remarked in earlier papers, there are essentially two sources of MusicXML scores. Firstly, there are exports made from scores in Finale, Sibelius etc. Such scores are typically created by amateur enthusiasts; they may contain mistakes and are not subject to any consistency or accuracy checks. Moreover, the various score-writing programs all generate different MusicXML for the 'same' music. Secondly, there is a large body of scores in Kern format, mainly from CCARH at Stanford. These scores are extremely accurate in their original format. However, the conversion to MusicXML is not good. Problems with scores have dogged our evaluations in previous years, so for the current campaign Donncha Ó Maidín undertook to check them carefully for syntactic and semantic errors such as incorrect elements, attributes, inconsistent bar numbering and so on.

Concerning the scores themselves, they range from 1 stave up to eighteen (Table 4). There are four symphonic movements by Mozart and Beethoven, the Berlioz Corsaire overture, and 'And the Glory of the Lord' from Handel's Messiah. These pieces range from ten up to eighteen staves. Next, there are two Vivaldi concertos, each on eight staves. Then there are two string quartet movements, by Haydn and Beethoven, on four staves, together with a Schubert song (Ständchen, D923) which is on three staves. Four Scarlatti sonatas, two Beethoven sonata movements, and pieces by Mussorgsky and Bartók lie on two staves and a movement from the first Bach Cello Suite (BWV1007) occupies a single stave.

All the scores are well-known pieces, so we were hopeful that suitable texts discussing those pieces would be available. This year, for the first time, we asked participants to set ten questions each. Firstly, the participant was asked to devise five questions, one each of types 1_melod, n_melod, 1_harm, n_harm, and texture (see Table 1 for example queries of these types). Secondly, the participant was asked to search likely text documents for noun phrases which could be used as queries. For example here is such a noun phrase about Bartók (but not about the Ten Easy Pieces in fact): 'two whole-tone pentachords A-E# and F#-C'[1]. We allowed minor modifications to the text for practical purposes, and we also permitted a bar restriction to be added (e.g. '...in bars 10-20') in order to reduce the search for matching passages in the scores. The participants did indeed set good questions and also returned well-formed answers, so this innovation was a success. Concerning evaluation, we decided to ignore the effect of knowing ten questions in advance, because all participants were in

[1] http://homepage.tinet.ie/~braddellr/disso/ch4.htm

the same position. Thus we did not eliminate from evaluation answers to the questions a participant had themselves set.

The remaining questions were set by the organisers. We tried to find suitable texts for the scores and then scanned these looking for suitable noun phrases. We then made up the balance with devised questions. In all, 49 queries were set from real texts (by participants or organisers) while the remaining 151 were devised by us.

Concerning the categories of query, the five basic types of previous years were used: 1_melod, n_melod, 1_harm, n_harm and texture. Examples can be seen in Table 1. A 1_melod is based on a single note while an n_melod is a series of notes of some kind. A 1_harm is a single chord and an n_harm is a series of chords. Finally, we have the texture classification for 'counterpoint', 'melody with accompaniment' etc. It is a basic classification but it has worked well for us.

As well the five basic categories, some of the queries are additionally assigned one of the types follow and synch (Table 2). These were also used in previous years, and allow more complicated queries. In essence, a follow query is one musical event coming after another ('continuo passage *then* a ripieno passage in measures 5-18'). In MIR terms, the juxtaposition of two phrases in this manner greatly reduces the number of matching passages, in the same way that word bigrams in Natural Language Processing are less frequent than the consituent words alone. This year, some follow queries had three phrases in sequence. A synch query specifies that two musical events are going on at the same time ('quarter notes C#, F# *during* a crescendo').

Following the pattern of previous years, queries were composed originally in ASCII Short Form and then converted into XML to make the Gold Standard. All questions and answers were checked by a second expert.

4 QUERY REPRESENTATIONS IN JSON

At the MediaEval 2016 Technical Retreat in Hilversum, The Netherlands, one possibility discussed was to divide the C@merata task into two stages: The first stage was to convert the natural language query into an intermediate representation which was closer to an Information Retrieval query; the second stage was to search the score using this derived information. The aim was to separate the NLP aspect of the task from the MIR. We decided to try this idea out. We built a multi-stage top-down parser which carries out the recognition and analysis of many kinds of musical textual phrase. The basic algorithm was from an earlier parser we built [3]. The output of this parser is then passed to the SpaCY statistical parser[2]. In the final stage, the output is analysed and a feature structure created. We chose JSON to represent the information as it is very flexible, well-known, and compatible with Python.

The various stages of the parser can be seen in Table 9. For each stage, an example input is shown together with the JSON output produced by the parser for that stage of processing. Each stage is concerned with one kind of constuct. For example, Stage 2 recognises that 'C Major' is a key and creates some attribute-value pairs to capture this information; similarly, Stage 19 recognises 'five-note melody' as a melody having a particular number of notes, this information being represented as further attribute-value pairs. Later on, a phrase like 'five-note melody in C Major' can be converted into a feature structure by combining the outputs of Stages 2 and 19. At the end, SpaCY parses the entire pre-processed input and can recognise any grammatical constructs, whether expected or not. However, the use of pre-processing considerably reduces the parsing ambiguity.

The parser was developed using the 2014-2016 data sets. It was then run on the 2017 queries and the resulting JSONs were added to the XML Gold Standard and made available to participants on request.

So far, this is work in progress. When we found constructs or concepts which were not being handled, we extended the representation accordingly. We carried out some error checking and made corrections along the way. However, there has so far been no formal evaluation.

Table 10 shows some examples of output for queries in the 2017 test set. As can be seen, quite complex constructs can be handled. We also have some provision for adding 'unknown' information into the JSON so that some downstream processor could carry out further analysis if need be. For example, 'a yearning melody in A flat' would be recognised with some certainty as a sequence of notes in a stated key. However, we could also note that a required property of the melody was that it was 'yearning', even though we might not know what that actually meant.

The aim of this work is to see what the limit is concerning the representation of a musical text, whether it is highly specific or quite vague. So far, we feel that a considerable amount of complex information can indeed be handled in this way but further detailed work is needed to take this further.

5 2017 CAMPAIGN

This year there were just two participants, CLAS and DMUN (Table 5). CLAS took part in 2014 and 2015 but was unable to undertake the task last year. DMUN has taken part in all four years of the C@merata task, but with a different leader for 2016 and 2017.

The CLAS system [11] was built using Python, NLTK, Music21 and MongoDB. Firstly, the query is parsed using NLTK, together with a feature-based Context-Free Grammar which specifies the controlled language for the C@merata music queries. This grammar was an extension of the one used in 2014 and 2015. The result is a feature structure corresponding to the key semantic elements of the query which is then used to retrieve results. The twenty MusicXML scores are indexed using MongoDB tables. For each score, there were four tables: Titles, Musical Events, Sequences and Analysis.

The query is answered by carrying out searches of the MongoDB tables using the query's feature structure, and

[2] https://spacy.io/

combining information derived from the results returned. A particular problem to be overcome was that feature unification as used in previous versions of the CLAS system can ignore attribute-value pairs, on one side or the other of the unification, which do not match. MongoDB queries do not have this property. Thus each feature structure derived from the original input query had to be converted into a NoSQL query to take account of this. Querying for sequences of notes in the database was performed by a series of searches, each checking if the event at the next timestep corresponded to the relevant sequence note. A similar approach was taken for chords and other sequences.

The DMUN system [2] takes as input a text query and initialises 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 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 of three core types of MIR function 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 comprises music elements, is converted into passages.

Each participant submitted one run. The results are shown in Table 6. According to the author of DMUN, most queries were processed with manual intervention and only a few queries were answered entirely automatically. Therefore the only automatic run is CLAS. The overall BF score for CLAS is 0.135 with a very similar MF of 0.166. Last year, the best scores were for DMUN01 (BF=0.070, MF=0.106) so this year is an improvement. Moreover, the task is definitely harder than last year and in addition DMUN in 2016 also declared some manual intervention and analysis on the basis of only a subset of attempted queries. So the CLAS result for this year is very good. CLAS scored very highly in 2014 (BF=0.797, MF=0.854) and 2015 (BF=0.620, MF=0.656). On the other hand, the task has increased greatly in difficulty; in the early years there were mostly simple notes etc. (F#, crotchet rest) and none of the advanced musical concepts and complicated syntactic structures we now have.

Table 7 shows the average results over both runs for different query types. By looking at the BF column, for example, we can gain some insight into the relative difficulty of the different types. Interestingly, 1_harm has the highest score (BF=0.286) followed by n_harm (0.255), n_melod (0.218), texture (0.158) and finally 1_melod (0.148). One would expect the 1_melod questions to be the easiest as they refer to simple individual notes. n_harm queries are sequences of chords and include cadences. Table 8 provides the same information but just for CLAS, this being the only fully automatic run. Once again, n_harm is the best (BF=0.269) followed by 1_harm (0.251), n_melod (0.151), texture (0.130) and finally 1_melod (0.076). This is once again surprising but could be accounted for by the fact that CLAS employed sophisticated chord and chord sequence processing, using in part the music21 *chordify* function, and building on their experience with the handling of chords in previous editions of C@merata.

Perhaps the 1_melod and n_melod were not that simple. 1_melod included 'G# at the start of a bar' where you need to know what the start of a bar means, and 'a dip down to the lowest note on the instrument' which is similarly rather poetic, and 'D two octaves lower than D4' which is a pitch relative to another pitch. For n_melod we had 'ten consecutive quavers in the left hand in bars 50-end' where you have to interpret 'ten consecutive' as well as knowing what 'end' is. Another example is 'four descending quavers' which involves knowing what 'descending means'. Of course, these vaguer and more figurative expressions are much more realistic and they show where natural language can come into its own as a means of specifying musical information. By contrast, simple, unambiguous concepts like 'C#' can be adequately handled in a symbolic query language suitably adapted to the music domain.

6 CONCLUSIONS

This year, there were several important innovations in the C@merata task. First, queries were contributed by participants for the first time. This involved a significant effort and commitment by the participants, for which we are very grateful. Moreover, they had to assimilate the finer points of the coding process such as the exact ASCII Short Form syntax, the procedures to follow in cases of ambiguity and the assignment of query-type tags like 1_melod and n_harm. However, it was a success and there was a significant gain in having composed queries contributed by new people who are moreover extremely expert in their own fields of music. They also extracted some very nice queries from real texts and along the way highlighted some important new sources for such texts.

As far as evaluation is concerned within a scenario where each participant knows some of the questions, we ignored this issue for 2017. There were only ten queries for each participant, out of 200, and indeed the participants could not necessarily answer their own questions anyway. The overall scores are low, so knowedge of 10/200 queries, i.e. 5%, does not seem to us a significant factor.

The second innovation was that more queries overall came from real texts than before – 49/200 with twenty contributed by participants and 29 contributed by the organisers. This number could be higher, but there is a significant difficulty in finding and analysing such texts. We need to devote more time to the collection of texts and their corresponding scores as an activity in itself. As always, we are hampered by the minimal supply of high-quality MusicXML scores.

Third, all the MusicXML scores were carefully checked this year for conformance to the standard in terms of elements, their context of use (relative to other elements), the attributes they have, and their values. For this invaluable work we must thank Donncha Ó Maidín who devoted a great deal of time to it. In previous years, Donncha has analysed MusicXML files from first principles using his own CPN software, while most participants tend to use Music21 [1]. This has given him a unique insight into the finer points. In addition to conformance, scores were also checked in respect of bar numbering. In MusicXML scores there can be problems in respect of anacrusis bars (which are conventionally numbered zero, but can be numbered one or even not numbered at all) and repeat bars (which can have a non-numerical number like 10a or, once again, no number). Bar numbering is extremely important to us as answer passages are identified in terms of them.

Fourth, we adopted points in the score in addition to the passages we have used since 2014. A point can capture an event which is instantaneous and which therefore does not involve a range of beats. Examples include cadences, where the change from the V chord to the I chord is the key defining aspect (and indeed the only one which is unambiguous), and changes of key signature or time signature. There were only a few such queries and, due to the low results overall, it is not possible to assess their effect reliably. However, this was useful work since points are clearly the correct way of answering certain questions.

Fifth, this year we once again used our five-way categorisation of scores – 1_melod, n_melod, 1_harm, n_harm and texture – and this worked well for us. No classification can work perfectly, but this one is largely correct for most queries and is therefore useful enough to be worth employing. A more detailed classification is much more complicated to work with for question encoders and is likely to display a larger number of shortcomings as well. In addition to the five types, we once again employed two modification types, follow and synch; loosely applied, these can characterise many of the more complicated kinds of queries because when there are two or more music events, either they are happening partly together or not together; if they are not together, one must be either before or after the other. Hence, follow and synch classifiers can capture many complex musical descriptions.

Sixth, we worked on capturing the content of a queries using a hierarchical feature structure which we expressed in JSON. Moreover, we wrote a parser for converting queries to JSON and this was developed in terms of the 2014-16 C@merata test sets. What we found was that this type of analysis was indeed possible, a high proportion of the semantic content of our queries could be captured in such a way, and that our parser, while far from perfect, was surprisingly good. This is work in progress, which took place in parallel with the C@merata evaluation, and more detailed and comprehensive work needs to be done on it. Also, an evaluation needs to be carried out. Such a representation can obviously not capture every subtlety of a query; consider the last example of Table 10: 'rocking eighth-note chords in the piano right hand against half-note octaves in the piano left hand in measures 1-10'. The JSON captures the length of the chords, the hand and the instrument playing them, the harmonic octaves and their length, the relationship between these two, and the bar restriction. It does not capture the vaguely-specified plurality of both components or the meaning of 'rocking'. The former can readily be addressed and we would propose that the latter kind of description could be captured by ad hoc attributes (e.g. additional_description) which could then be processed by a downstream component if required. There are always going to be vague and ambiguously specified aspects to a musical description, and we need a way of working with these within a more specific type of feature structure.

Turning now to the overall conclusion for the task, the CLAS result was very good considering the difficulty, but there were insufficient participants and hence runs to get a reasonable spread of results which could be analysed properly.

Finally, concerning work which we can do in future, there are several possibilities. First, we can derive more queries from real texts; we need to work on this more systematically, and over a longer time period than the C@merata task itself. Second, we could expand our passage representation to pick out answers more precisely; at present, we have vertical 'lines' through the score but we have no 'horizontal' ones – we do not identify which stave(s) contain the answer and also we do not know which parts within a matching stave are relevant. For example, there could be two horn parts on the one stave, but only one of those could match the answer. Ways of doing this have been suggested in other contexts [10] which we might be able to adopt. Third, we can work more on the JSON representations as already discussed, and fourth we can consider widening participation by further parameterisation of the task.

REFERENCES

[1] Cuthbert, M. S., & Ariza C. (2010). music21: a toolkit for computer-aided musicology and symbolic music data. Proceedings of the International Symposium on Music Information Retrieval, Utrecht, The Netherlands, August 09 - 13, 2010, p637-642.

[2] Katsiavalos, A. (2017). The DMUN System at the MediaEval 2017 C@merata Task. Proceedings of the MediaEval 2017 Workshop, Trinity College Dublin, Ireland, September 13-15, 2017.

[3] Sutcliffe, R. F. E. (2000). Using A Robust Layered Parser to Analyse Technical Manual Text. Cuadernos de Filología Inglesa, 9(1), 167-189. Número Monográfico: Corpus-based Research in English Language and Linguistics. http://www.csis.ul.ie/staff/Richard.Sutcliffe/murcia_parsing_paper00_repaginated.pdf

[4] Sutcliffe, R. F. E., Collins, T., Hovy, E., Lewis, R., Fox, C., Root, D. L. (2016). The C@merata task at MediaEval 2016: Natural Language Queries Derived from Exam Papers, Articles and Other Sources against Classical Music Scores in MusicXML. Proceedings of the MediaEval 2016 Workshop, Hilversum, The Netherlands, October 20-21, 2016. http://ceur-ws.org/Vol-1739/MediaEval_2016_paper_55.pdf.

[5] Sutcliffe, R. F. E., Crawford, T., Fox, C., Root, D. L., & Hovy, E. (2014). The C@merata Task at MediaEval 2014: Natural language queries on classical music scores. Proceedings of the MediaEval 2014 Workshop, Barcelona, Spain, October 16-17 2014. http://ceur-ws.org/Vol1263/mediaeval2014_submission_46.pdf

[6] Sutcliffe, R. F. E., Crawford, T., Fox, C., Root, D. L., & Hovy, E. (2014). Shared Evaluation of Natural Language Queries against Classical Music Scores: A Full Description of the C@merata 2014 Task. Proceedings of the C@merata Task at MediaEval 2014. http://csee.essex.ac.uk/camerata/.

[7] Sutcliffe, R. F. E., Crawford, T., Fox, C., Root, D. L., Hovy, E., & Lewis, R. (2015). Relating Natural Language Text to Musical Passages. Proceedings of the 16th International Society for Music Information Retrieval Conference, Malaga, Spain, 26-30 October, 2015. http://ismir2015.uma.es/.

[8] Sutcliffe, R. F. E., Fox, C., Root, D. L., Hovy, E., & Lewis, R. (2015). The C@merata Task at MediaEval 2015: Natural language queries on classical music scores. Proceedings of the MediaEval 2015 Workshop, Dresden, Germany, September 14-15 2015. http://ceur-ws.org/Vol-1436/Paper12.pdf.

[9] Sutcliffe, R. F. E., Fox, C., Root, D. L., Hovy, E. and Lewis, R. (2015). Second Shared Evaluation of Natural Language Queries against Classical Music Scores: A Full Description of the C@merata 2015 Task. Proceedings of the C@merata Task at MediaEval 2015. http://csee.essex.ac.uk/ camerata/.

[10] Viglianti, R. (2015). Enhancing Music Notation Addressability. http://mith.umd.edu/research/project/enhancing-music-notation-addressability/.

[11] Wan, S. (2017). The CLAS System at the MediaEval 2017 C@merata Task. Proceedings of the MediaEval 2017 Workshop, Trinity College Dublin, Ireland, September 13-15, 2017.

Table 1: Query Types

Type	No	Example
1_melod	26	G# at the start of a bar a dip down to the lowest note on the instrument half-note C in the left hand in the treble clef the highest note in the vocal line
n_melod	75	semiquaver melody G D B A B D B D ten consecutive quavers in the left hand in bars 50-end arpeggio-like passage in the right hand in measures 1-12 series of eighth notes in the right hand, starting on an off beat
1_harm	30	Db triad in the right hand dominant seventh broken chord of the key of C five-note chord quarter-note chord F# C# A# E in the whole orchestra in measures 150-180
n_harm	47	six consecutive sixths in the right hand in bars 1-25 a cadence, G to C in measures 7-10 chord of F major, chord of C major, chord of F major double stopping on two successive notes in the bass
texture	22	monophony first five notes of the fugal entry commencing at bar 27 melody with accompaniment in measures 1-6 homophonic passage in measures 164-170
All	**200**	

Table 2: follow and synch Queries within 1_melod, n_melod, 1_harm and n_harm

Type	No	Example
follow	19	three thirds followed by three crotchets followed by three thirds in the left hand in bars 1-43 melody Bb C Eb Bb A followed by a Db chord six repeated chords E G B C# followed by six repeated chords F# C# A# continuo passage then a ripieno passage in measures 5-18
synch	14	triplets against even eighth notes in measures 1 to 10 quarter notes C#, F# during a crescendo four descending sixteenth notes in the strings against a chord in the winds in measures 190-199 cellos and basses leading into the shadows while the upper strings accompany with gently throbbing harmonies in measures 73-87

Table 3: Scores Used. 'Origin 2014' means the score was previously used in 2014.

Work	Staves	Scoring	Lang	Origin
bach_cello_suite_1_bwv1007_prelude	1	vc	Eng.	2014
scarlatti_sonata_k30	2	hpd	Eng.	2015
scarlatti_sonata_k281	2	hpd	Eng.	2016
scarlatti_sonata_k320	2	hpd	Eng.	2016
scarlatti_sonata_k466	2	hpd	Amer.	2014
bartok_10_easy_pieces_n4_sostenuto	2	pf	Amer.	2017
mussorgsky_pictures_promenade_m1	2	pf	Eng.	2017
beethoven_piano_sonata_14_op_27_no_2_m1	2	pf	Amer.	2017
beethoven_piano_sonata_14_op_27_no_2_m2	2	pf	Amer.	2017
schubert_staendchen_d923	3	T, pf	Amer.	2017
beethoven_string_quartet_op_18_no_3_m1	4	2 vn, va, vc	Amer.	2017
haydn_string_quartet_no_57_op_74_no_1_m1	4	2 vn, va, vc	Amer.	2017
vivaldi_concerto_4_vn_rv580	8	4 vn, 2 va, vc, db	Amer.	2016
vivaldi_conc_vn_op6_no6_rv239_m1	8	3 vn, va, vc, db, hpd	Eng.	2016
mozart_symphony_no40_m4	10	fl, 2 ob, 2 bn, 2 hn, 2 vn, va, vc, db	Eng.	2016
beethoven_symphony_no_1_m1	12	2 fl, 2 ob, 2 cl, 2 bs, 2 hn, 2 tpt, timp, 2 vn, va, vc, db	Amer.	2017
beethoven_symphony_no_4_m1	12	fl, 2 ob, 2 cl, 2 bs, 2 hn, 2 tpt, timp, 2 vn, va, vc, db	Amer.	2017
beethoven_symphony_3_movement_iii_muse	13	2 fl, 2 ob, 2 cl, 2 bs, 2 hn, 2 tpt, timp, 2 vn, va, vc, db,	Eng.	2016
berlioz_corsaire_overture_h101	17	fl, 2 ob, 2 cl, 4 bs, 4 hn, 2 tpt, 3 trbn, tuba, timp, 2 vn, va, vc, db	Eng.	2017
handel_messiah_and_the_glory	18	fl, 2 ob, cl, bs, hn, trbn, tuba, SATB, hpd, 2 vn, va, vc, db	Eng.	2016

Table 4: Distribution of Scores by Number of Staves

Staves	Frequency
2	6
3	1
4	6
5	2
8	2
10	1
13	1
18	1
All	**20**

Table 5: C@merata Participants

Runtag	Leader	Affiliation	Country
CLAS	Stephen Wan	CSIRO	Australia
DMUN	Andreas Katsiavalos	De Montfort University	England

Table 6: Results for All Questions: The DMUN01 was experimental and used manual intervention on all but 2-3 queries. Thus CLAS01 is the best automatic run. BP=Beat Pecision, BP=Beat Recall, BF=Beat F-Score, MP=Measure Precision, MR=Measure Recall, MF=Measure F-Score.

Run	BP	BR	BF	MP	MR	MF
CLAS01	**0.099**	**0.212**	**0.135**	**0.122**	**0.260**	**0.166**
DMUN01	0.833	0.155	0.261	0.924	0.172	0.290
Maximum	0.833	0.212	0.261	0.924	0.260	0.290
Minimum	0.099	0.155	0.135	0.122	0.172	0.166
Average	0.466	0.184	0.198	0.523	0.216	0.228

Table 7: Average Results by Question Type: BP=Beat Pecision, BP=Beat Recall, BF=Beat F-Score, MP=Measure Precision, MR=Measure Recall, MF=Measure F-Score. Note that **follow** and **synch** questions are across **1_melod**, **n_melod**, **1_harm** and **n_harm**.

Type	BP	BR	BF	MP	MR	MF
1_melod	0.355	0.230	0.148	0.448	0.320	0.192
n_melod	0.550	0.167	0.218	0.569	0.190	0.239
1_harm	0.727	0.178	0.286	0.727	0.178	0.286
n_harm	0.482	0.218	0.255	0.632	0.254	0.310
texture	0.588	0.103	0.158	0.588	0.103	0.158
follow	0.534	0.026	0.044	0.567	0.040	0.063
synch	0.000	0.000	0.000	0.000	0.000	0.000

Table 8: CLAS Results by Question Type: BP=Beat Pecision, BP=Beat Recall, BF=Beat F-Score, MP=Measure Precision, MR=Measure Recall, MF=Measure F-Score.

Type	BP	BR	BF	MP	MR	MF
1_melod	0.043	0.328	0.076	0.062	0.475	0.110
n_melod	0.137	0.167	0.151	0.176	0.213	0.193
1_harm	0.636	0.156	0.251	0.636	0.156	0.251
n_harm	0.250	0.290	0.269	0.263	0.304	0.282
texture	0.176	0.103	0.130	0.176	0.103	0.130
follow	0.067	0.026	0.037	0.133	0.053	0.076
synch	0.000	0.000	0.000	0.000	0.000	0.000

Table 9: JSON Analysis: Stages of Processing

No	Stage	Example Input	JSON
1	**texture**	two-part texture	`{ 'texture': 'two_part' }`
2	**key**	C major	`{ 'key_name': 'C', 'key_accidental': 0, 'key_type': 'major' }`
3	**note**	slurred double whole note trill	`{ "note_divisions": 48, "note_length": 384, "note_ornament": "trill", "note_performance": "slurred" }`
4	**note_sequence**	C#4 D4	`"note_sequence": [ { "note_accidental": 1, "note_name": "c", "note_octave": 4 }, { "note_accidental": 0, "note_name": "d", "note_octave": 4 } ]`
5	**measure**	bars 1-10	`{ "measure_from": 1, "measure_to": 10 }`
6	**underlay**	on the word "Der"	`{ "note_underlay": "Der" }`
7	**instrument_quote**	"Cello"	`{ "instrument": "Cello" }`
8	**staff**	left hand	`{ 'staff_hand': 'left' }`
9	**instrument**	violin I divisi	`{ "instrument": "violin", "instrument_direction": "divisi", "instrument_group": 1 }`
10	**instrument_sequence**	cellos and double basses	`{ "instrument_list": [ { "instrument": "cello" }, { "instrument": "doublebass" } ] }`
11	**triad**	Ib triad	`{ "relative_pitch": 1, "triad_inversion": 1 }`
12	**interval**	doubly diminished harmonic fifth	`{ "interval_augmentation": -2, "interval_harm_melod": "harmonic", "interval_size": 5 }`
13	**interval_sequence**	alternating fourths and fifths	`{ "interval_list": [ { "interval_harm_melod": "harmonic", "interval_size": 4 }, { "interval_harm_melod": "harmonic", "interval_size": 5 } ], "interval_seq_pattern": "alternating" }`
14	**cadence**	interrupted cadence	`{ "cadence": "interrupted" }`
15	**inversion**	in the first inversion	`{ "triad_inversion": 1 }`
16	**chord**	chord of F#3, D4 and A4	`{ "chord_word": true, "note_sequence": [ { "note_accidental": 1, "note_name": "f", "note_octave": 3 }, { "note_accidental": 0, "note_name": "d", "note_octave": 4 }, { "note_accidental": 0, "note_name": "a", "note_octave": 4 } ] }`
17	**arpeggio**	F sharp minor arpeggio	`{ "arpeggio_word": true, "key_accidental": 1, "key_name": "F", "key_type": "minor" }`
18	**scale**	C major scale	`{ "key_accidental": 0, "key_name": "C", "key_type": "major", "scale_word": true }`
19	**melody**	five-note melody	`{ "melody_word": true, "note_count": 5 }`
20	**by**	See examples below	
21	**simultaneous**	See examples below	
22	**loose_indication**	fermata on a whole note	`{ "note_divisions": 48, "note_length": 192, "note_performance": "fermata" }`
23	**time_signature**	12/8	`{ "time_higher": 12, "time_lower": 8 }`

Table 10: Examples of Queries Converted to JSON

dotted crotchet Bb in the right hand in bars 23-40

```
{
   "first": {
      "measure_from": 23,
      "measure_to": 40,
      "note_accidental": -1,
      "note_divisions": 48,
      "note_length": 48,
      "note_length_multiplier": 1.5,
      "note_name": "b",
      "note_octave": -1,
      "staff_hand": "right"
   },
   "second": {},
   "type": "simple"
}
```

dotted crotchet chord B2 B3 D#5 in bars 1-46

```
{
   "first": {
      "chord_word": true,
      "measure_from": 1,
      "measure_to": 46,
      "note_divisions": 48,
      "note_length": 48,
      "note_length_multiplier": 1.5,
      "note_sequence": [
         {
            "note_accidental": 0,
            "note_name": "b",
            "note_octave": 2
         },
         {
            "note_accidental": 0,
            "note_name": "b",
            "note_octave": 3
         },
         {
            "note_accidental": 1,
            "note_name": "d",
            "note_octave": 5
         }
      ]
   },
   "second": {},
   "type": "simple"
}
```

monophonic passage lasting twelve crotchet beats

```
{
   "first": {
      "note_divisions": 48,
      "note_length": 48,
      "number": 12,
      "texture": "monophony"
   },
   "second": {},
   "type": "simple"
}
```

G# quaver in the right hand against a crotchet in the left hand in bars 1-25

```
{
   "first": {
      "note_accidental": 1,
      "note_divisions": 48,
      "note_length": 24,
      "note_name": "g",
      "note_octave": -1,
      "staff_hand": "right"
   },
   "second": {
      "measure_from": 1,
      "measure_to": 25,
      "note_divisions": 48,
      "note_length": 48,
      "staff_hand": "left"
   },
   "type": "against"
}
```

descending arpeggio in quavers followed by ascending arpeggio in quavers in bars 1-30

```
{
   "first": {
      "arpeggio_word": true,
      "direction": "falling",
      "note_divisions": 48,
      "note_length": 24
   },
   "second": {
      "arpeggio_word": true,
      "direction": "rising",
      "measure_from": 1,
      "measure_to": 30,
      "note_divisions": 48,
      "note_length": 24
   },
   "type": "followed_now"
}
```

rocking eighth-note chords in the piano right hand against half-note octaves in the piano left hand in measures 1-10

```
{
   "first": {
      "chord_word": true,
      "instrument": "piano",
      "note_divisions": 48,
      "note_length": 24,
      "staff_hand": "right"
   },
   "second": {
      "instrument": "piano",
      "interval_harm_melod": "harmonic",
      "interval_size": 8,
      "measure_from": 1,
      "measure_to": 10,
      "note_divisions": 48,
      "note_length": 96,
      "staff_hand": "left"
   },
   "type": "against"
}
```