CRIM 2020 Development Tasks

From the 2020 Development Document

Similarity

Grounding Similarity in Music.

- We are interested in “similarity in music”, which stands behind almost any attempt to explain *style* (which is about what works of a certain “sound” share at the highest level), *genre* (which binds pieces by purpose or pattern) and *borrowing* (when one piece quotes, reworks, or otherwise borrows from another specific piece). Even our attempts to show how a single piece holds together (or not) hinges on our capacity to show how its motives, harmonies, or gestures are

"similar" or "not." Musicologists have long been concerned with these questions, but how can machines be used to advance such work? Can they show us more about . .

Match Model with Derivative.

- This is our corpus' primary information: each of its Masses was created in 'direct reference' to one pre-existent model. Any machine we create should first be able to master the basic task of telling us which model goes with with derivative.

Ranking Similarity.

- Can we find degrees of similarity among pieces? Aside from those pieces bound by explicit Model-Derivative, what would subsequent degrees of similarity look like?
    - Similar *soggetti* (certain melodic-rhythmic patterns)? Can these be found via HumDrum or similar tools? Via some direct search of MEI? Via some search of tokenized MEI?
    - Similar *counterpoint* (certain interval patterns, or 'interlocks'). See Three-Grams noted below. These could be found in the MEI, or some representation of it.
    - Similar *presentation* types (via metadata, the time or intervals of entry)
    - Similar *procedures* (via relationship type metadata; pieces that share some *habits* of quotation, transformation, etc).
    - *Economy and Consumption*. How much of a model is used? How much of a derivative is new or borrowed? These would also tell us something about how works are like each other? Perhaps this would also include some measure of *where* the borrowed material appears in each piece (example: some mapping of borrowings according to beginning/middle/end of each piece).
    - The metadata would also tell us about similarity within and beyond a given piece:
        - Simply Identity: The same EMA and same Metadata: its the same passage, but only if the Derivative and Model are the same in all instances!

- Self Similarity: Same Piece, Different EMA but the same Metadata: these are similar passages in the same composition.
- Exo Similarity: Different pieces (and of course different EMA) but the same metadata. These could be:
    - Model and Mass (that is, some example of borrowing)
    - Any Pair of pieces _not_ otherwise known to be Model and Mass. These share some stylistic or proceduraly similarity!

Specific Techniques and Tools.

jSymbolic

- Which could be applied as:
    - pieces as Bag-of-Notes (including the Movements of Masses as individual pieces in this case)
    - phrases as Bag-of-Notes (using CRIM Phrase data to segment the phrases, perhaps relying in information about words and rests in the melodies)
    - various other Bags such as date, place, genre of model, or genre of Mass Movement

Network Graphs using CRIM metadata

- examples such as DRB's Network of PEns, based on sequences of time or melodic entries of voices. others?
- possibly networks of soggetti, based on tokenized representations of melodic and rhythmic motion

Linguistic Tools and Bioinformatics Tools

- such as Levenshtein Distances. DRB has already experimented to show which Observations are most similar to each other, based on one such routine using tokenized representations of melodic intervals of entry.
- now also see Janco music21 patterns; how to apply various 'similarity' measures to these?

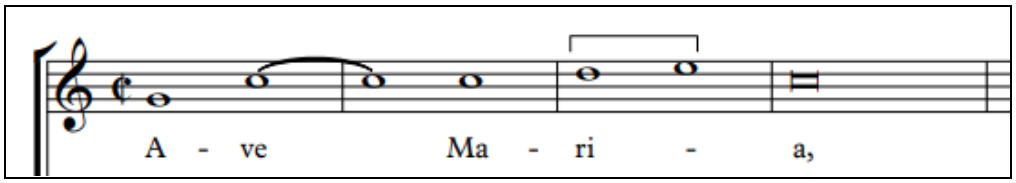
Pattern Finding

- ELVIS technique of three-grams at the level of the minim involving modules of two voices. https://github.com/ELVIS-Project/vis-framework.

- Here they represent a fundamental motion in a lower part by (a ‘vector’ of movement up or down by diatonic interval X) surmounted by a pair of vertical harmonic intervals (for instance a sixth followed by a third). This ‘triad’ represents the basic element of contrapuntal motion. Any succession of voices can be represented as an interlocking chain of such three-grams. Julie Cumming can put us in touch with students who know the system

Janco music21 patterns and n-grams

- Notebook with all Code
- Archive of pre-Slack Correspondence between RF and AJ
- music21 and intervall patterns
    - We are using music21 to read the MEI and calculate intervals using notesToInterval. This gives us an interval object with a name attribute. For example, P5. We are dropping the character and keeping the number.
    - All note-to-note intervals are represented by a number.
        - When the melodic interval is ascending the value is positive or negative for descending.
        - The numeric value is followed by a letter with the rhythmic interval between the notes. L shows the duration of the second note is longer than the first. S shows the duration of the second note was shorter. And E notes that the duration of the two notes is equal. There are three types of intervals with rests in them.
        - An interval with a note followed by a rest has a melodic interval of 0. The same is true of a rest followed by a note.
        - An interval of two rests is R, followed by the rhythmic interval.
    - E to F is 2
    - F to E is -2
    - E to rest is 0
    - rest to E is 0
    - E (duration 4) to F (duration 4) is 2E (Equal)
    - E (duration 4) to F (duration 8) is 2L
    - E (duration 8) to F (duration 4) is 2S
    - Rest to Rest is R (with same system for rhythmic duration)
    - For example, the first four measures of Ave Maria [Superius] are:



- <note xml:id="m-64" dur="1" dur.ges="1024p" oct="4" pname="g" pnum="67" stem.dir="up">
- <note xml:id="m-65" dur="1" dur.ges="1024p" oct="5" pname="c" pnum="72" stem.dir="down">
- <note xml:id="m-83" dur="1" dur.ges="1024p" oct="5" pname="c" pnum="72" stem.dir="down" />
- <note xml:id="m-84" dur="1" dur.ges="1024p" oct="5" pname="c" pnum="72" stem.dir="down">
- <note xml:id="m-98" dur="1" dur.ges="1024p" oct="5" pname="d" pnum="74" stem.dir="down">
- <note xml:id="m-99" dur="1" dur.ges="1024p" oct="5" pname="e" pnum="76" stem.dir="down" />
- <note xml:id="m-118" dur="breve" dur.ges="2048p" oct="5" pname="c" pnum="72" stem.dir="down">
- <mRest xml:id="m-138" dur="breve" />

start	start_duration	start_id	end	end_duration	end_id	interval
G	4	m-64	C	8	m-64/m-65	4L
C	8	m-64/m-65	C	4	m-84	1S
C	4	m-84	D	4	m-98	2E
D	4	m-98	E	4	m-99	2E
E	4	m-99	C	8	m-118	-3L
C	8	m-118	rest	8	m-138	0E

- These intervals can then be used as an array: ['4L', '1S', '2E', '2E', '-3L', '0E']
- Here is a list of all intervals in the CRIM corpus: crim_intervals.csv

Experiment 1: “N-grams”

In natural language processing and computational linguistics, n-grams are a common tool for the study of text. An n-gram is a segmentation of the text into units of size n. We could take “Ave Maria” for example. Each character in the string is a 1-gram (unigram) including spaces and punctuation. In “Ave_Maria” we get 9 unigrams.

We can also split the string into eight bigrams:

Av, ve, e_, _M, Ma,ar,ri,ia

or 3-grams:

Ave, ve_,e_M,_Ma,Mar,ari,ria

Up to a single 9-gram

This same concept can be applied to the CRIM interval arrays:

['4L', '1S', '2E', '2E', '-3L', '0E']

Interval sequence bigrams:

['4L', '1S'],['1S '2E'],[‘2E’, '2E'], [‘2E’,'-3L'] [‘-3L’, '0E']

Interval sequence trigrams:

['4L', '1S', '2E'], ['1S', '2E', '2E'], ['2E', '2E','-3L'],['2E','-3L','0E']

Interval sequence 4-grams

['4L', '1S', '2E', '2E'],['1S', '2E', '2E','-3L'], ['2E', '2E','-3L', '0E']

With n-grams we are able to create a list of all of the patterns of various length that appear in the CRIM corpus, It gives us a list of all existings patterns. We can then count the occurance of each pattern in the corpus.

Here is a link to a CSV of these patterns and their frequency in the corpus:

crim_pattern_counts.csv

The spreadsheet begins with sequence bigrams given that a single interval is not likely to be significant for identifying patterns across scores or parts. The longest n-gram is a 101-gram that appears twice in the corpus. All n-grams that appear less than twice were dropped.

Please note I have removed all-rest patterns from the list. There are many places where a singer rests for 10 or 12 measures, but this is not an event of interest in the score.

Similarity

As an initial experiment, I tried using the n-grams as a measure of similarity between pieces. For each part in each score, I create an intervals array. I can then identify the n-grams present in the array. It's hardly a "fingerprint," but it does give us a common unit of measure that can be compared across scores. I have approached this as a set operation. Each part has its own set of n-grams. In theory, the union of two sets will be larger for similar scores and smaller in less similar scores. I'm not thinking of this as an end result, but rather a way of sorting the pieces so that we can identify scores that are relatively more likely to be connected in some meaningful way. How we assess that connection and identify its specific components is an open question.

In addition to counting which n-grams are present in a part, I am counting the number of times that an n-gram appears in the part. This information could be used to improve our measure of similarity. For example, two pieces are only similar when they share both an n-gram and a similar frequency of that n-gram. We could adjust the required count similarity as needed. I'd be happy to add this specificity if it would lead to improvement in our measure of similarity.

One possible way forward is to create an n-gram search and visualization application. With a given interval n-gram or set of n-grams, we could identify similar scores and visualize them in such a way that a music scholar could interpret and make sense of the machine's observations. It could be something like "auto-suggest" for CRIM observations.

I have also been experimenting with Palladio as a way to investigate and interpret the data. Using a graph, for example, we can visualize which n-grams various scores have in common. In the image below, we see six clusters of n-grams at the intersections of four scores (Sanctus, Ave Maria, Missa Baisez moy, and scores with no title). If the intersections of scores lend useful information, we could try to use something like UpSetR to visualize them.

Here is the sorted lists of parts by "similarity"

Crim_ similarity.csv

Experiment 2: Melodic "Phrases"

Upon re-reading a message from Rich, I have shifted to try a different approach to segmenting the interval arrays. Rather than chopping them up into all possible

combinations, I am cutting them on the 0 intervals. We then have a segment of melody that ends with a rest.

Here is a link to my code for this "phrase" approach.

Here is the resulting file, a list of the interval "phrases" as they appear in CRIM sorted by their frequency. Note that each pattern is given a name for simplicity.

Crim_phrase_patterns.csv

A version of the same file, but without 1 intervals. Because repeated notes are often used to break up time to accommodate new syllables in a text. And when composers borrow one melody for another text, they are forced to divide or combine notes.

crim_phrase_patterns_no1s.csv

Experiment 3: Quantized Phrases

- Many times our melodies have various ornaments or passing tones. What if we try to compare them using some grid of 'structural' tones as determined by some minimum duration. That is: quantizing the melodies will abstract them, allowing us to compare the fundamental movement:
- See music21 method quantize. If we then test quantized melodies phrases using the approaches in Experiments 1 and 2, what are the results?
- Would these abstractions necessarily lose their XML:IDs? Could we preserve some other information about the location of these tunes relative to the source pieces? Measure numbers as a range? (And then generate EMA with "1-10/@all/@all" addresses?)

Experiment 4: Text Phrases

- CRIM tells us where "first" and "last" bar in which any line of text is heard: http://crimproject.org/pieces/CRIM_Model_0001/.
- These are also available as *CRIM Phrases*: http://crimproject.org/admin/crim/crimphrase/, which turn could in turn easily be rendered as EMA references to slice the relevant MEI. The EMA for measures 1-10 of a piece would be ". . . .1-3/all/@all"

- We could use this as another way to segment CRIM MEI data (and thus avoid the problem of short or intermedial rests, and to allow for motives created by text repetition to be part of the story of similarity.
- The text phrase method would also be of use in Experiment 5, below.

AJ June 1 Notebook here

With Music21, we can access the lyrics associated with any note in the score (note.lyric). As we calculate intervals, I have added a start_lyric and end_lyric field to the interval dictionaries. We can then iterate over all of the intervals and use that information to splice the list into sequences based on the lyrics.

This approach does present some trouble given that many of the lyric phrases have long sections where the melody changes, but the lyric does not. For example 'gra - - ti -a ple - - - - - na," (measures 8-12, *Ave Maria*, [Superius]) Where the hyphens make the sustained lyric clear in the notation, in music21 we have ['gra-', None,'-ti-','-a','ple'. None x 5, '-na,'] How can the machine know that a None in the sequence is not the end of the phrase? Looking at the page, these sustained melodies are still broken by rests. Borrowing from earlier work where we break on 0 intervals, I look for an interval of 0 that has no lyrics. That is the end of the lyric phrase. So whenever we have a start_lyric, we find the end of the phrase where start_lyric is None, end_lyric is None and there is a 0 interval. This seems to work, though it would be nice to be more deliberate in the logic.

Slight update, I compared the note intervals to the lyric intervals. The lyric intervals includes rest-rest intervals that were incorrect. The lyric phrases ends before the rest-rest intervals. I have now corrected the code and updated the results below:

🎤 lyric: [('A-', '-ve'), ('Ma-', '-ri-'), ('-ri-', None), ('-a,', None)] (0, 6)

🎵 notes: [('G', 'C'), ('C', 'C'), ('C', 'D'), ('D', 'E'), ('E', 'C'), ('C', 'rest')] (0, 6)

🎤 lyric: [('gra-', None), ('-ti-', '-a'), ('-a', 'ple-'), ('ple-', None), ('-na,', None)] (7, 18)

🎵 notes: [('C', 'B'), ('B', 'A'), ('A', 'G'), ('G', 'A'), ('A', 'G'), ('G', 'C'), ('C', 'B'), ('B', 'A'), ('A', 'B'), ('B', 'C'), ('C', 'rest')] (7, 18)

🎤 lyric: [('-mi-', '-nus'), ('-nus', 'te-'), ('te-', None), ('-cum,', None)] (20, 36)

🎵 notes: [('C', 'C'), ('C', 'D'), ('D', 'E'), ('E', 'C'), ('C', 'B'), ('B', 'A'), ('A', 'G'), ('G', 'F'), ('F', 'E'), ('E', 'C'), ('C', 'B'), ('B', 'A'), ('A', 'G'), ('G', 'F'), ('F', 'E'), ('E', 'rest')] (20, 36)

🎤 lyric: [('Vir-', '-go'), ('-go', 'se-'), ('se-', '-re-'), ('-re-', None), ('-na,', None)] (37, 43)

🎵 notes: [('C', 'A'), ('A', 'G'), ('G', 'C'), ('C', 'D'), ('D', 'E'), ('E', 'rest')] (37, 43)

🎤 lyric: [('-re-', None), ('-na.]', None)] (46, 49)

🎵 notes: [('C', 'B'), ('B', 'C'), ('C', 'rest')] (46, 49)

🎤 lyric: [('A-', '-ve'), ('-ve', 'cae-'), ('cae-', '-lo-'), ('-lo-', '-rum'), ('-rum', 'Do-'), ('-mi-', '-na'), ('-na', None)] (50, 60)

🎵 notes: [('C', 'C'), ('C', 'D'), ('D', 'E'), ('E', 'E'), ('E', 'F'), ('F', 'E'), ('E', 'E'), ('E', 'D'), ('D', 'E'), ('E', 'rest')] (50, 60)

Experiment 5: Contrapuntal Arrays

- Can we begin go think about counterpoint? One way would be to look at our melodic (and or rhythmic) motion in more than one voice at once.
- Arrays could be based on Phrase data (which would capture musically meaningful combinations of contrapuntal lines). Of course we know that some of the measure ‘boundaries’ for phrases overlap, and so there will be some extra information in these phrases. But such overlaps will still make musical sense from the standpoint of counterpoint.
- One melodic phrase might look like this:
    - `['0E', '2E', '2E', '2E', '0E']`
- An “array” of contrapuntal lines might look like this:

    - `[['0E', '2E', '2E', '2E', '0E'], ['0E', '2E', '2E', '2E', '0E'],['0E', '2E', '2E', '2E', '0E'],['0E', '2E', '2E', '2E', '0E']]`

- Arrays could be tested for 'similarity' based on shared patterns of motion: "-5" in the lowest part can only be accompanied by a limited number of intervallic distances in the parts above it.

- Perhaps we could also use music21 to calculate the *harmonic* intervals among the voices (something like an intervallic array from the lowest voice to the highest) and sample these at some rate. This would yield pieces that *sound the same*, as opposed to those that used the *same motives in different ways*.

Experiment 6: ELVIS Applied to Multivoice Pieces

- McGill's ELVIS system offers another interesting way to understand *countrapuntal patterns* between voices. It represents movement as a series of 3-grams:
    - for the lower voice: the melodic distance from one note to the next
    - for upper voice: the harmonic interval that this voice makes against each of the notes in the lower part
    - code here (but it needs work) Julie Cumming @ McGill has promised that students there can help

- But how to run ELVIS against CRIM multivoice pieces? A four voice piece would in fact involve multiple ELVIS streams of 3-grams, as we track the motion between S-A, S-T, S-B, A-T, A-B, T-B. That is SIX ELVIS streams! What do we do with the results?

- Would it make sense to segment by CRIM Text Phrases? Or?

Results to date

- 3-grams PNG
- All NGrams (without segmentation by rest)

Gould Notebooks June 9 Status Report

- See Zoom Recording of AJ, RF, and FG conversation.

- Collaboratory now includes all features in one space: https://colab.research.google.com/drive/1ZB14nLwQgjbQ-udTfLskliY70vs0vem6?usp=sharing
    - Parse MEI and music21
        - Main method gets as many MEI files as requested
        - Parses MEI as music21, returning lists of notes for each piece. These are defined as 'notes 1' 'notes 2' etc.
        - It is also possible to define a range of measures, or a specific number of measures
        - >Score | Voice Part | Start Measure | Stop Measure
        - options for range:
            - specific range
            - all parts (voices) in range
            - all parts (voices) in all ranges
    - Vectorize music21
        - as semitones (zero indexed)
        - as generic diatonic intervals (3, not -3 or +3)
    - Algorithms for Similarities and Matches
        - System looks for edit distances between similar or exact strings of vectors.
        - User can set threshold of number of vectors (ex: "5")
        - Rests serve to 'break' the string (**1, 2, R, 3, 4**, for ex, would simply be *ignored* altogether at a threshold of 5. But **1 2 R 3 4 5 6 7** would return **3 4 5 6 7**)
        - For similarity ranking: If one item differs in a string, then score of "1", etc. In short: "how many changes would be needed to make these the same." Then ranks them. So this method would allow us to capture 'flexed' soggetti.
    - Next Steps
        - Rhythmic information
            - Parse data as 1/4 note duration reference (this is the most granular, and built in to music21). This will allow us to run similarity string searches *exact* proportions (or rhythmic 'distances') Up 50%, down 25%, and so on. The 'floating number' approach is most exact
            - Then we can generalize the exact durations as *relative* movement, such as Longer, Shorter, Equal.
            - Andy's system used number for a melodic vector and a letter for a rhythmic one. But python lists allow us to use any kind of data.
        - Data Model in Python?
            - Should the results be modelled as a dictionary, object, or data class?
            - We need for each stream or string
                - Title of piece

    - Voice(s)
    - XML IDs (in the original MEI)
    - Measure Ranges (for the EMA

Find My Phrase (or Counterpoint) Idea

- Normally in CRIM we have been thinking about ways to discover similar melodies automatically. The guidance we offer includes:
    - Choice of corpus or pieces
    - Diatonic or Chromatic melodic intervals (generic or semitones)
    - Length of the 'set' of vectors
    - Sampling rate (all notes, or by offset or by beat)
- Could we also consider the problem *in reverse?* That is, the user provides a particular phrase, and asks "how many phrases are similar to *this*?" The machine looks for them and provides the answer. The same thing might even be done for ELVIS-style contrapuntal n-grams: "how many passages are similar to *this*?"
    - The pseudo code might look like this:
        - user selects notes in CRIM
        - user copies the EMA (along with URL of the piece)
        - the full 'address' of the passage is passed to OMAS, which returns valid MEI consisting of *only* the selected notes
        - The MEI is in turn converted to music21
        - the music21 is vectorized or treated to the VIS system
        - search for that pattern in the corpus (perhaps it's been pre-processed in some way to speed this up)
- Questions:
    - How account for the fact that these examples would be of varying length? Would it matter if they were longer than the 'typical' vector-similarity window? Or would we check the length of the vector and then search within that window?
    - What would determine the length of the matches? Some moving window of similarity?

Exemplifier: Output Formats of music21 Results

- Currently the 'array' of results is returned in the terminal as various sets of related items, each with interval vectors, measure ranges, piece names, sets of durations.
- These are, moreover, ranked in terms of degrees of identity and similarity (based on how many "steps" would be needed in order to transform the given result into an "exact" match.
- Would it be possible to translate these results into a CSV file or similar workspace that would contain things like:

        - the melodic Vectors
        - the rhythmic durations
        - PieceID
        - PieceTitle, et
        - Measure Ranges
        - the EMA
        - Some (arbitrary) ‘heading’, of the sort that Andy showed in one of his first experiments, in which each group is given a random word as its title
        - The EMA concatenated with the piece URL and OMAS service, and so that it was ready to be ‘sliced’ as valid MEI and then returned via Verovio to the user? This would allow rapid testing of the results. Or perhaps the results could be rendered directly in the notebook this way.
    -

Discrete Fourier Transformation

- See Summary of readings and discussion with Harding and Moss.

Genetics and Bioinformatics