

David Lewis, Frans Wiering

Practicalities of Corpus Building

Creating and Exploring Digital Data

Abstract

In this panel, we present a number of approaches to the creation of music corpora, using manual and fully- and partially-automated methods, and we consider the effects these approaches have on the nature, size and data encoding of the respective collections. We also explore how the process is affected by the nature of the source materials used. We illustrate the impact of these approaches for later use, such as simple web or paper publication or searching and analyzing the newly-available musical information.

Introduction

Corpora have long held a central role in the use of computers for music – from the substantial editions compiled by the Princeton Josquin Project in the 1960s and 1970s to the searchable incipit catalogues of RISM today. There are many reasons for developing large musical collections – they can facilitate statistical analysis, they can enhance accessibility by distribution via a single source, and they can prove more cost-effective than their print equivalents, especially for the later incorporation of corrections. Today's corpora are more varied than ever, as this session demonstrates – they may even be virtual corpora, discovered as found images on the internet; but many of the concerns identified as early as 1966¹ remain applicable. How do we get the music into the computer? How much musical information should we supply? What other information do we provide? How should we check for errors? How many errors are acceptable? To what uses will the corpora be put? What information that can be gleaned from the music in a corpus is useful? How should we present searches and findings?

The papers and discussion that follow do not set out explicitly to answer these questions, but they do grapple with them.

In the context of a larger discussion about music encoding in general, and of MEI in particular, there are some initial observations to make, to which we return in our later discussion. None of the projects described below uses MEI or any other interchangeable or standard encoding schema for its internal working. Some of the projects have early support for MEI as an interchange format, and there are intentions to expand this.

Finally, one theme that runs through most of the projects described below is interaction – primarily through the Web – in the form of online dissemination, editing and searching. This is a relatively recent development for digital music corpora, and is proving the most significant driver of innovation and enhanced access, changing the relationship between user and resource and, arguably, putting the user in a more prominent and powerful position than ever before.

David Lewis, Jeffrey Dean & Ronald Woodley, *Mixing Text and Music: Editing Music Treatises for a Digital Edition of Tinctoris*

Traditional editions of musical materials are extremely labour intensive, and represent a considerable investment in time, energy and money from their editors, typesetters and publishers. Whilst one might imagine that a modern age of computers and the Internet would reduce these costs – and other articles from this panel give examples of where this can be the case – it is far easier to reduce the effort of distribution than that of the editorial task itself. Given the extra potential for richness and complexity offered by a digital edition, it is even possible for the level of detail expected of modern editors to greatly exceed that of their print-based antecedents.

¹ Barry S. Brook (ed.), *Musicology and the Computer; Musicology 1966–2000: A Practical Program* (New York, 1970)

As a result, it can become as important to focus on the needs of the contributors when preparing software for a new editorial project as on those of the future audience. As long as the information from the edition is accessible to future developers, other interfaces for users can be designed, but the time spent in preparing the edition is harder to repeat.

In this article, we will be discussing the tensions between user-centric approaches that focus on editors, readers and developer communities, in the context of a project editing medieval treatises on music. We also explore the difficulties of catering for an unknown potential user base, looking at some strategies for handling them and the pitfalls inherent in these approaches.

The Complete Theoretical Works of Johannes Tinctoris: A New Digital Edition

Johannes Tinctoris (c.1435–1511) is an important figure for musicologists researching the fifteenth century. As a writer, he completed fifteen treatises on music, of which twelve survive today, and these works are given authority by his status as a musician and composer and his familiarity with his most important musical contemporaries. His writings range from highly technical discussions of notational minutiae to broader aesthetic and historical commentaries, and so they are of wide interest to scholars. Musical illustrations abound, and vary from the simple – even just a single symbol – to substantial pieces of polyphony.

The complete treatises have been edited before², but the edition is in need of updating and correction, and the texts have never been translated in their entirety into English.

In Spring 2011, the UK Arts and Humanities Research Council announced that it would fund a project, directed by Ronald Woodley, to prepare a digital complete edition of Tinctoris' writings, including Latin texts and English translations, all the musical examples, and to present these alongside modern commentary material on the text. The edition is to be made available online, and the tools developed to support the creation of the edition to be freely distributed. Although there is a wider advisory panel, all editorial duties fall to Jeffrey Dean and Ronald Woodley.

Pilots and the Original Proposal

The project proposal was based on the preparation of online editions of three treatises between 2000 and 2003 by Woodley.³ These editions were shown in a frame-based website, allowing users to view a treatise's structure, its Latin text and an English translation in parallel. Musical examples were prepared using a graphics package to compile glyphs from a bespoke font. References and commentary are linked together in a way that reflects the sort of ambitions of McGann's concept of the Hypertext⁴.

It was clear that this way of producing music examples, although it produces very high-quality typesetting, is not ideal from two perspectives: it is time-consuming to typeset and edit; and the results are opaque, with no possibility of automated exploration of the results, which would allow for searching or music information retrieval applications. The alternative that we explored was to use a notationally-sophisticated score editor to generate images for the web pages and, at the same time, standards-compliant musical files.

During a one-year pilot project preceding and accompanying our main funding application, we extended Gsharp – a free, open-source score-editing program, which has some history of use in Music Information Retrieval (MIR) applications – to provide support for the historical notations used by Tinctoris in his treatises, and also to make the software more intuitive for new users.

Over the course of the pilot project, the software was extended to the point where all the music examples that had been edited in the previous iteration of the project could be replicated within the editor. Where Gsharp had previously been controlled entirely by keyboard commands, dialogue boxes, menus, icons and limited drag-and-drop functionality were added.

2 Tinctoris, Johannes, *Opera theoretica*, ed. Albert Seay, 2 vols. plus vol. iia (Corpus scriptorum de musica, 22; n. p., 1975; vol. iia, Neuhäusen-Stuttgart, 1978).

3 Funded by the then Arts and Humanities Research Board, Leverhulme Foundation, the Worshipful Company of Musicians and Birmingham Conservatoire (part of Birmingham City University)

4 Jerome McGann, 'The Rationale of Hypertext', in *Electronic Text. Investigations in Method and Theory*, ed. Kathryn Sutherland (Oxford, 1997), 19–46.

What was not attempted within the pilot project, since there was no funding for it at the time, was to create any new treatise editions from scratch. It was with this, after the new project started, that issues began to emerge.

Tinctoris incorporates music in his treatises with fluency, moving from text to music in a continuous flow, sometimes with only a few words separating examples. Switching between applications, though it had seemed reasonable in our proposal, was an annoying distraction. This problem was intensified by the usual problem of the development cycle. If an editor found that a notational element was not yet implemented in Gsharp, he would have to report the issue and wait for it to be resolved before the music could be entered. Depending on the nature of the absent feature, the matter might be resolved in moments, or it might take weeks, during which time all examples containing the notation would have to be skipped – something that could easily cover most of the examples in a treatise.

In such a case, the order of editing would be dictated not by musicological argument, or availability of sources, but by technological convenience. Especially given our discussion earlier about the primacy of editors and their time, it is clear that this is not a satisfactory state of affairs.

At the same time, we saw an opportunity in a single-application paradigm for a richer encoding style, covering both the text and the music, and allowing for some of the information available to TEI editors to be used. The most important element of our application design, however, was that it should suit the immediate needs of our limited user base, rather than trying to cater for a possibly imaginary larger audience of editors. This is not to deny the usefulness of designing software for general use, merely to reflect a few points: firstly, that, without substantial allowance for testing and development cycles with a representative number of users, such an audience would remain little better than imaginary, and no funding is in place for that scale of work; and, secondly, that our main aim was to create a particular corpus, and whatever software was needed for it – other goals, though desirable, are secondary.

Yet Another Textual Input Language

The approach we chose to remedy these issues was to create a textual input language from scratch, supporting text, music, annotations and basic cataloguing information. The language was devised primarily by Jeffrey Dean, and is optimized for the particular task of manually entering these treatises. Whilst we would expect it to work equally well in other contexts, that is not our main purpose in devising it. Future transformation to TEI/MEI is important for us, and the design of the language has been moderated by reference to the TEI and MEI models.

The language itself is parsed and processed in JavaScript in a browser to give either a live-rendering editor or, on the public pages for the edition, various views of the finished text. The parser itself is constructed to skip parts of a provided encoding that it cannot satisfactorily process, thus allowing the editor to add new features directly into their edition as soon as they notice the need. Until the changes are added to the software, the feature will usually be invisible, but no modification will be required to the encoding once they are implemented.

Some Features of the Language

Although giving details of the language itself is perhaps unnecessary – we make no claims for its generality of use, at least not at this stage – we believe that explaining a few features of the language will help provide a rationale for its development and inform the discussions below. The language is relatively informal, and relatively compact, with readability and ease of typing generally preferred over regularity and parser maintenance.

In general, mark-up that applies to a region is given using the familiar, SGML-style element tags, which are required to be balanced. Other types of information generally use other forms of brace. A simple example of the latter is the use of multiple punctuation methods in our edited texts.⁵ Where there are options available, they are indicated by the editor enclosed in curly braces, as here:

```
<verse>Maxima{.,} longa{.,} brevis{.,}
semibrevis{.,} minima.</verse>
```

⁵ Users can choose to read the edited Latin texts with a punctuation style that reflects contemporary practice or one that reflects later innovations but is perhaps more familiar.

Music examples are given with their own tags and prefatory material, followed by the common note shape/staff position scheme. In the example below, two longs (notes two levels higher than the semibreve in duration) are given in normal and inverted form, on the upper and lower spaces respectively of a three-lined, clefless staff:

```
<example: {in-line}, {mensural: void}, {staf: 3, red}>
{clef: 0} {solm: 0} {mens: 0} L7 -L5</example>
```

Where a clef is needed, a pitch-based code can replace the numbers given in the example above.

Several elements of the encoding resemble macros, designed to make code that is abbreviated without sacrificing too much readability. Elements whose angle braces are replaced by curly ones apply for one item only and require no explicit closure (e.g. {full}La is equivalent to <full>La</full>). In a significant departure from SGML-derived languages, tag-indicated hierarchies may, under certain circumstances, overlap, to reduce the need for user-provided IDs or the artificial early closure of a tag.

The most significant opportunity offered by the change of approach to our software was that it allowed the editors to add text-critical mark-up in a way that would not have been possible in the sort of static edition originally envisaged. The language supports textual and musical variants in the same way, and allows editors to describe the nature of the variant and indicate the associated sources, using a shorthand (e.g. *Ligatura* {var="autem" V BU Br1 : (om.) G} descendens). Variants within variants, in both music and text, are supported using the same syntax – just a simple nesting:

```
[...]quod note minores ipsam maiorem imperficientes valorem
eius non attingant{.,} {var="ut etiam probatur in hoc
{var="sequenti" BU : (om.) V} exemplo" V BU : "ut hic" Br1 :
"ut in exemplo sequenti" G}:...
```

User Interfaces

Although moving from a graphical input method to a textual one may appear to be a retrograde step, it came initially from a specific user need – the editor could more easily see how to perform the task at hand with a keyboard than with mouse alone or keyboard and mouse. This is something that we would expect to vary from user to user. In a larger project, it is possible that either multiple input systems or some sort of hybrid would be necessary, but in our case it was sufficient to build something that satisfied our immediate requirements. This change also simplified the addition of information that is often difficult to handle in graphical applications. The interface presented to an editor is a simple, two-pane view, with a textual input box on the right and the rendered edition on the left (see figure 1). Although originally updating live, the edition is now redrawn only on command. This is partly because redrawing is slow relative to typing (largely due to our heavy use of SVGs for music examples), and partly because small changes in the text can have large, non-local effects – the resulting jumping around of the edition can prove distracting.

Direct graphical editing of the rendered treatise is possible, and graphical palettes similar to those used in the Electronic Corpus of Lute Music (see below) have been tested, but this mix of interaction methods has not yet been found to be useful.

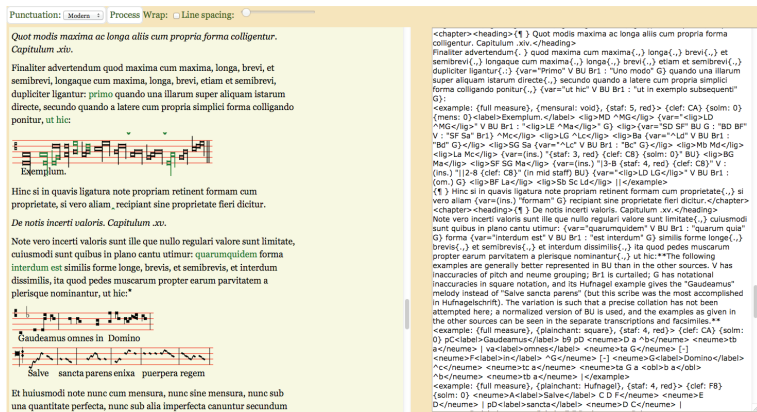


Figure 1: The editor's view, with the rendered edition (left) generated from the contents of the text field (right).

For readers consulting the edition, the encoding is, of course, invisible. Readers visiting the web site can view any completed treatise either as it appears in its manuscript sources, in edited form or in translation, singly or in parallel on screen (see figure 2). Because we use the SVG format for our music examples, these are fully scalable and produce high-quality print output.

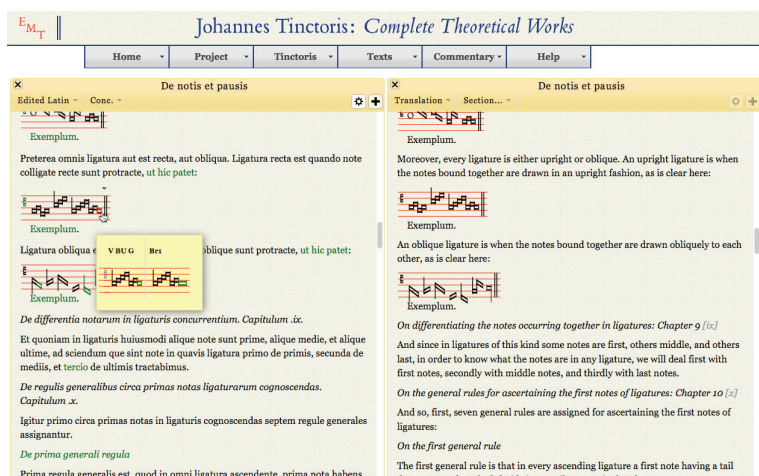


Figure 2: A visitor's view, with edited Latin on the left and a translation on the right. Variants are displayed in green and, when the mouse hovers over them, expand to show details as shown.

Clearly, we have some ideas of how some users will consult the edition, but it is equally clear that there will be much that we will fail to anticipate, and much that will be unintuitive or awkward in our first iterations. Where the creation and development of the input language could be shaped directly by responding to the (small) user base, this is much harder to achieve with a website that is freely available online. The creation of a blog (<http://earlymusictheory.org/blog>) and the hosting of discussions on it is intended to go a certain way towards soliciting opinions and requests from as large an audience as possible. For less formal or rich discussion, a twitter feed (#earlymusictheor) has been created, and attracted almost a hundred followers even before any edited texts were made available online. We hope, with time, that these followers will themselves become more active contributors to our discussions and consultations.

Standards and Legacy

The encodings and the parsing and rendering software are made freely available both through the project website and on GitHub,⁶ but our approach does raise questions around a more active view of software sustainability. Unless our on-line community asks for it, we have no plans to turn our special input language into

6 <https://github.com/BirminghamConservatoire/JohannesTinctoris>

something more generally used and, though it will be fully documented, we have no current intentions to run tutorials in its use for corpus building. This is a long way from our original goal of building an intuitive editor for mixed early music notations.

This may be too negative a view. If either the wider community were to choose to embrace this way of creating editions – something we hope to discover through discussion on our blog – or if we simply view the method we have used as an effective scaffolding for building the corpus, it will be a success even if not, of itself, one of lasting importance.

Whatever the success of this approach, our priority, after ensuring that the encoding process is efficient and that the editions can be prepared within the limited time available, is that the treatises themselves be accessible to all kinds of users, and that includes the MIR community and information discovery systems themselves. The first step for this, making the rendered editions as clear and semantically sound HTML and SVG as possible, allows crude searching and exploration. Anything more sophisticated would require a parser to interpret our markup policies, however consistent they may be.

Of more value, both for this project and for the community, would be to tweak the parser to output and parse TEI/MEI. Most of the components needed are already present, although there certainly is both more detail in our encoding and a wider spread of notational features in the source treatises than is specified in the current standards. As yet, there is no software that would be capable of meaningfully processing the result of such an encoding, but it remains the most likely way of creating a sustainable resource – especially if we can contribute code towards the development of general parsing software for TEI/MEI-encoded documents.

Gsharp, the graphical editor, also has potential to be useful here, but extending it to become an editor of text with musical inserts is well beyond our resources within this project.

Discussion

The advantages that our approach offers for our project are clear: the editing process can be carried out in a relatively fast, uninterrupted way; software can be developed and improved speedily, smoothly and continuously; there is unlikely to be a point at which the richness of the edition is below the level intended by our editors. The disadvantages – all arising from the use of a bespoke, non-compatible, only partly intuitive encoding language – can be addressed by giving attention to the way in which our edition is presented, through visualizations and TEI/MEI export.

One of the benefits of the increased power of a modern computer is that developing such complex and rich systems can be seen as a relatively nimble solution to a specific set of requirements, rather than a costly folly, whilst the development of open encoding schemas means that we can, we hope, share our information without the substantial extra effort that would have been necessary a relatively short time ago.

Tim Crawford, *Early Music Online and The Electronic Corpus of Lute Music*

The Electronic Corpus of Lute Music (ECOLM)⁷, has been funded by the UK's Arts and Humanities Research Council at various times over the past one and a half decades (1999–2006, 2011–2012). Its overall goal is to provide a machine-readable corpus of European lute music, in recognition of the fact that this is a large and important repertory of music, representing over 300 years of music history (roughly 1480–1800), possibly amounting to some 50,000 different pieces in total. Furthermore, it is generally little understood because of its idiosyncratic notation, lute tablature, a prescriptive notation system that gives the player of the instrument a 'recipe' of instructions concerning string-fret 'coordinates' for each note and the relative timing of sequences of simultaneous groups of notes (which may be 'chords' or single notes depending on the musical texture).

Tablature does not describe musical features such as note-pitches (which depend on the tuning of the instrument), voice-leading or individual note-durations; these must be the subject of interpretation by the player or of algorithmic or heuristic processing of an encoded tablature. It is, therefore, essential that the encoding captures as much of the original graphical appearance as is possible or at least feasible within the pragmatic

⁷ www.ecolm.org

constraints of an ASCII encoding scheme originally developed in the late 1980s. In principle, it should always be possible to render a ‘diplomatic’ version of the tablature from the machine-readable code (using suitable screen fonts, if necessary produced from original images of historical printed tablaturs) which can be used by a modern lute-player. The code used, TabCode, is described briefly in *TabCode for Lute Repertories*⁸, and documented on the ECOLM web-site⁹.

As well as making available a considerable inaccessible repertory which is little understood by musicians in general, ECOLM in principle allows comparison between lute music and musical repertory in standard notation, given the possibility of extracting compatible features from the encodings; for example, it is possible to extract sequences of ‘harmonic’ features (such as chromagrams) from both forms of notation which can be used for alignment in much the same way that audio and MIDI files can be aligned.¹⁰ This will lead to an enhanced awareness – in the context of computational musicology, at least – of the historical significance of lute music.

In the early phases of ECOLM’s development, all TabCode encodings were done manually, with an inevitably high cost in terms of human-expert time for training, data-input and proof-correction. It was soon recognised that building a genuinely useful resource in a reasonable time would demand automatic encoding methods. For printed lute tablaturs, such a method has been developed by Christoph Dalitz as an add-on (referred to in this document as OTR, for Optical Tablature Recognition) to the Gamera document analysis framework,¹¹ for which TabCode output was added in a recent phase of ECOLM funding. This works well on the products of many 16th-century printers of lute tablature, allowing rapid data capture from suitable digitised page-images. The OTR is carried out as an offline process, and the output stored in the ECOLM back-end SQL database system.

In 2012, funding was obtained for ECOLM to work with images and metadata from a new resource, Early Music Online (EMO), which contains page-images from around 25 lute tablature prints within its total of 324 printed books of music from the 16th century held at the British Library. EMO, a joint project with the BL and RISM UK under the direction of Dr Stephen Rose (Royal Holloway, University of London), hopes soon to be able to expand to include all of the more than 2,000 printed music books from the Renaissance period which have been acquired by the library over several centuries.

Data-capture for the vocal music in EMO is being done within the current ECOLM project using the mensural-notation recognition software, Aruspix, as described in Laurent Pugin’s contribution to this article. The resulting ‘automatic’ encodings can be used in similar ways to those in ECOLM, with the advantage that, since they derive from individual part-books rather than scores, melodic strings can be relatively easily produced that may be used to search musical data resources such as RISM, for example.¹²

One important reason for ECOLM to carry out work not only on the lute tablaturs but also on the vocal-music part-books in EMO is the considerable overlap between them due to the fact that about half the 16th-century printed lute repertory comprises arrangements of vocal music for solo lute.¹³ For approximately 1,000 of the solo lute pieces identified in EMO before the project began, the breakdown looked like this:

8 Tim Crawford, ‘TabCode for Lute Repertories’, in *Computing in Musicology: A Directory of Research*, vol. 7, 1991, pp. 57–62

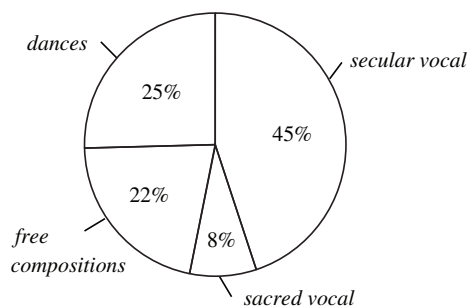
9 www.ecolm.org

10 Dannenberg, Roger B. and Ning Hu. (2003), ‘Polyphonic Audio Matching for Score Following and Intelligent Audio Editors’, in *Proceedings of the 2003 International Computer Music Conference* (San Francisco: International Computer Music Association, 2003) pp. 27–33

11 <http://gamera.informatik.hsnr.de/addons/otr4gamera>

12 <http://opac.rism.info>

13 Howard Mayer Brown, *Instrumental Music printed before 1600: A Bibliography* (Cambridge, Mass.: Harvard University Press, 1965)



ECOLM Online Correction

Although the output from good digital images is very accurate, OTR is not perfect, just like any optical recognition system. The knock-on effect of most individual errors in tablature recognition is generally not as serious, for example, as the anomalies that can sometimes be caused by incorrect recognition of a single clef, note-length or accidental in optical recognition from scores in conventional notation. However, despite the fact that a few lost percentage points in accuracy may not reduce the usefulness of an encoding for a player (used to reading from poor facsimiles of badly-written manuscripts), it is important to represent the state of the source as accurately as possible. The ECOLM project enlisted the help of the UK Lute Society to provide willing correctors, for whom an interactive online correction interface has been developed which presents, aligned on screen, a single randomly-selected line of tablature from the original source-image and its corresponding modern tablature rendered from the 'raw' TabCode stored in the ECOLM database.

The screenshot shows the ECOLM Online Correction interface. At the top, there are logos for the Electronic Corpus of Lute Music, Arts & Humanities Research Council, and Goldsmiths University of London. Below the logos, there is a navigation bar with 'No. 2 allocated Tue Sep 4 2012 3:02pm', 'Hide previous assignments', and 'timg | Change password | Log out | Help'. The main area displays a comparison between original lute tablature (top) and modern tablature (bottom). The modern tablature is rendered from TabCode. The interface includes a play button, a 'Show TabCode' button, and a 'Submit corrections' button.

The TabCode from which the modern tablature is rendered is updated interactively within the browser; once a corrector has completed editing the modern tablature to match the original image for a line of music, it is submitted to the ECOLM database and a new line of tablature presented for correction. The interface allows correctors to do as much or as little in a session as they wish; the database keeps track of the current state of their effort, so that a full 'Undo' trail is automatically maintained at all times when the browser is connected to the internet.

Unless they specifically ask to see it (using the 'Show TabCode' button on the interface) the TabCode is hidden from correctors, with the intention that all necessary editing operations can be done through the interface. From the earliest days of TabCode, the possibility to embed comments in Pascal-style curly brackets has always been present; in some cases, these can be structured, as in the case of an 'invisible symbol' such as a system- or page-break (encoded as "{^}" and "{>}", respectively). In the TabCode output by the OTR software,

we redundantly store detailed information about each glyph recognised, basically its TabCode and its location on the page-image. Since our online editor never alters comments, at any editing stage we can thus recover both the original (uncorrected) code and the current edited version. Here is what happens (schematically) when a spurious glyph is deleted from the TabCode by the interface:

before:

```
glyph1_code{glyph1_code: glyph1_location}
glyph2_code{glyph2_code: glyph2_location}
```

after:

```
{glyph1_code: glyph1_location}
glyph2_code{glyph2_code: glyph2_location}
```

This can be used to evaluate the approximate recognition error-rate of the encoding, as we can directly compare the 'raw' and corrected TabCode very easily. Since we are having each system corrected by two (different) people, we can further compare the performance of correctors. Such comparisons could be done according to the original source, or the output of a certain 16th-century printer, or even by the time of day at which the corrections were done (all are timestamped).

While we have not attempted a comprehensive evaluation, which we intend to do at such a time when meaningful statistics have been gathered and the results can be fed back directly into improvements in the OTR system, we present here a simple example of a variation in recognition error-rate that we had not anticipated *a priori*.

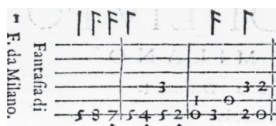
We found that the highest rate of OTR accuracy was obtained on a set of images from a 1562 edition of music by the most famous 16th-century lutenist, Francesco da Milano (1497–1543), edited by his pupil, Perino Fiorentino (1523–55), and printed by Antonio Gardane in Venice.^{14, item 1562,}



50% of the 70 systems (bearing on average 221 separate tablature glyphs) from this book which had been double-corrected at the time of writing were considered perfect – that is, no changes needed to be made by either corrector. However, the overall error rate was consistently found to be higher (1.41%) for the first system of each piece in the book than for the others (0.4%) Why should this be? The answer lies in the typographical layout adopted by this particular printer, which presents each piece-title, printed sideways, at the left end of the first line of music. Not surprisingly, the OTR software attempts to read this as tablature, with nonsense results:

14 Howard Mayer Brown, *Instrumental Music printed before 1600: A Bibliography* (Cambridge, Mass.: Harvard University Press, 1965)

Original:



OTR output:



Overall, we observed error-rates as follows, which reflect the fact that this problem is by no means restricted to this particular book:

All systems: **1.69%**
 First systems: **2.39%**
 Other systems: **1.60%**

However, it should be stressed that these impressively low error-rates (still necessitating a good deal of correction effort) may be unrealistic, since we consciously began the OTR work on sources which we considered likely to produce good results. There are several tablature books which are printed extremely badly or from worn type, or which only survive in copies in poor condition; error-rates will inevitably be higher in such cases.

Laurent Pugin, *Early Music Online and the Challenges of Encoding Part-Books*

The Early Music Online (EMO) is a project that was initiated through a JISC digitization project involving the British Library, the Royal Holloway University of London and the RISM UK group. Through this undertaking, 324 music books held at the British Library were digitized and the images were made freely available online¹⁵. The music books are printed anthologies from a wide range of countries, including Italy, France, and Germany, representing a total of about 10,000 pieces of music. Most of the collection is vocal music, but it also contains music for instruments, including keyboard or plucked instruments. The books were digitized from microfilms, in grayscale. This made it possible to have the project realized very rapidly and at reasonable cost and to produce at the same time images suitable for research. One special aspect of the EMO project is that in addition to images it provides rich metadata about the books. The metadata includes information about the printers and the locations where printing took place, but also about the composers of each piece. It includes lists of the content of each book. This information, together with the digitized images, constitutes an extremely valuable resource for researchers.

Launching EMO was a first step in digital scholarship for sixteenth-century music. However, with images and metadata being made available online, there is still no possibility for the scholars to search the music content directly. Making the content of such collection searchable can be achieved through optical music recognition (OMR), and this is what the ECOLM III project¹⁶ experimented with. Whilst the main focus of ECOLM III was to transcribe lute music from EMO via optical tablature recognition (OTR), it also partly dealt with OMR on mensural notation. The latter was performed with the Aruspix software application, a tool specifically designed for early typographic music prints. Aruspix is an adaptive software application, which means that it can learn and improve itself dynamically when correct pages become available. The corrections are made via a dedicated music editor that makes it easy for the user to spot recognition errors and to correct them. Aruspix uses MEI as its encoding scheme.

15 S. Rose: "Early Music Online," *Early Music Performer*, 30, pp. 22–25, 2012.

16 Tim Crawford, see <http://gtr.rcuk.ac.uk/project/15588947-3835-4FF8-AA8B-7BD124BB546D>

During the ECOLM III project, a global evaluation of what can be achieved with the current existing OMR technology was performed. A global evaluation on such a large and diverse collection is of utmost importance for setting up large-scale projects because it gives indications of what is feasible. The idea was to use the EMO collection to evaluate two key aspects in corpus building. The first one is to know what percentage of books can reasonably be processed with the OMR technology we currently have, knowing that any large collection will always contain atypical books that cannot be processed. Early printed music books, such as those of the EMO collection, can be difficult to process because of the book format, the music notation type, or the printing technique that was used. In order to be able to estimate this percentage, all the books of the EMO collection were screened considering three criteria: the book format, the notation type and the printing technique. The second aspect evaluated was learning, for the books that can be processed, what baseline recognition rate we can reasonably expect to obtain with current technology. To answer this question, one page of each of the books that can be processed was fully corrected by hand and then used to calculate the recognition rate the tool produces.

Book Format

Most of the books of the EMO collection are part-books, the standard format of the time. In part-books, each part (voice or instrument) is printed in a separate book, or gathering. The EMO collection also contains choir books, where all the voices are also printed separately, but in the same book, on the same page (or pair of pages) for the singers or musicians to be able to sing or play together by sharing the book. Finally, the EMO collection contains ten table books (Figure 3) that were mostly printed by Jacques Moderne in Lyon (e.g., K.10.a.9)¹⁷. In a table book, all the voices or parts are printed on the same page as in a choir book but not all in the same direction. They were intended to be laid on a table with musicians sitting on different sides of the book. The table books in EMO contain parts printed upside-down.

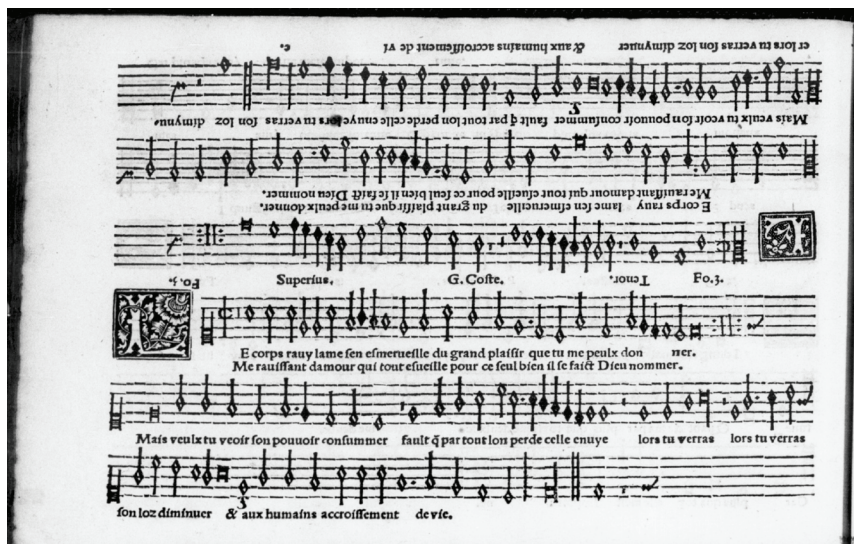


Figure 3: A table book with one part printed upside-down. The book was printed by Jacques Moderne in Lyon in 1541.

Notation Type

The large majority of the books of the EMO collection are in mensural music notation, on five-line staves. This was the most common notation during the sixteenth century. The collection contains twenty-seven books of lute tablature (e.g., K.1.c.11) and five books of organ tablature (e.g., K.8.h.22). There are six interesting books that contain a mix of mensural notation and lute tablature (e.g., D.250 and K.2.h.12). There is also one book that contains a very rare case of a mixture of five-line mensural notation and four-line square notation, printed by Etienne Gueynard in Lyon in 1528 (K.9.a.23); see Figure 4.

¹⁷ The books are identified by their signature at the British Library. The Images can be accessed from the RISM UK website (www.rism.org.uk).

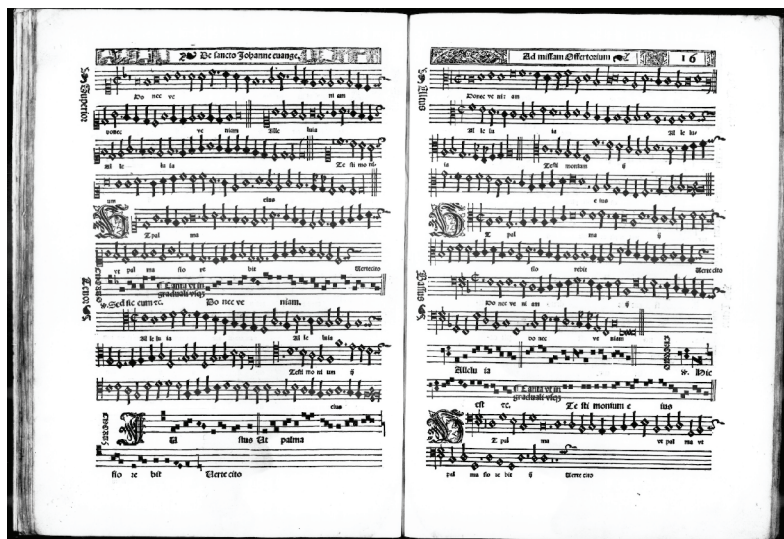


Figure 4: A very rare case of a print with five-line mensural notation mixed with four-line square notation. The book was printed by Etienne Gueynard in Lyon in 1528.

Printing Technique

In the EMO collection, nearly all of the books (about 300) were printed using the single-impression typographic technique. This technique was pioneered by Attaignant, who introduced an innovative way of printing staff lines and notes in one single pass using new typographic types. The EMO collection also contains books printed early, by Petrucci, who used a multiple-impression technique where staves, notes and text were printed in three successive passes. The collection also contains seven books printed from plates engraved by Simone Verovio, one of the first music engravers; see Figure 5. A further five books from EMO are in woodcut, printed mostly by Andrew Antico (e.g., K.8.b.7).

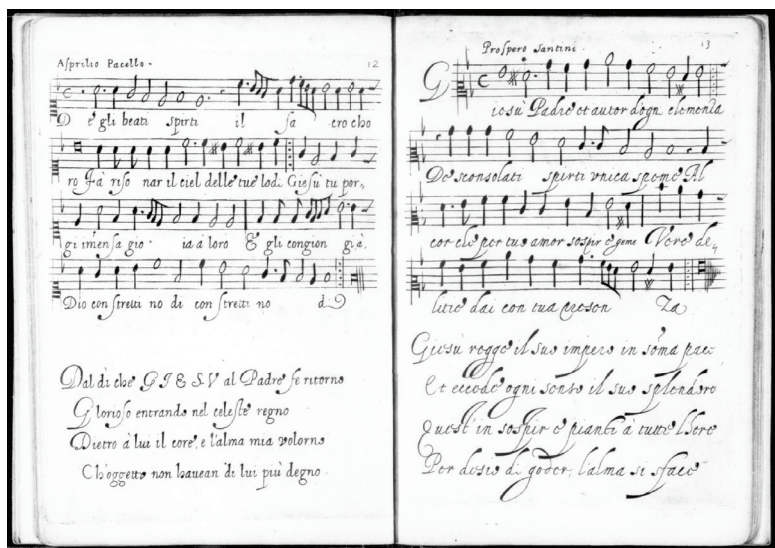


Figure 5: A book printed from engraved plates from 1591. The book was printed in Roma by one of the very first music engravers, Simone Verovio.

What Can We Obtain with Current OMR Technology?

By taking into account the book format, the notation type and the printing technique for all the books of the EMO collection, we were able to determine which books could reasonably be processed with the Aruspix software application. These include all of the books in mensural notation printed with single-impression

technique – the notation type and printing technique for which Aruspix was developed. We left out the books using multiple impressions because Aruspix has shown inconsistent performance on this type of print, mainly because text or ornate letters are sometimes superimposed on the music staves. Lute tablatures and books printed from engraved plates were also left out because none of these can be processed with Aruspix. Finally, the table books and the book with a mix of mensural notation and square notation were also left out.

Out of the 324 books of the EMO collections, 260 books remained, which means that about 80% of the collection can appropriately be processed with Aruspix. It was interesting to notice that the books were printed by nearly 50 different printers coming from 17 different cities and two unknown locations. Looking at the provenance of the books more precisely revealed that it reflects quite well the printed music production of the time and the importance of the different cities as shown in Figure 6.¹⁸

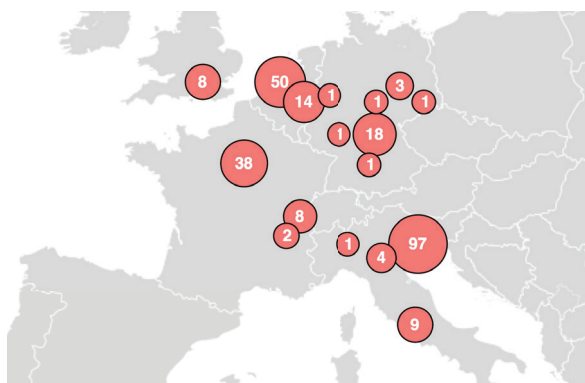


Figure 6: In the EMO collection 260 books have an appropriate format, notation type and printing technique to enable them to be processed with Aruspix. They were printed in 17 different cities, reflecting the importance of the music printing centres of that century.

Looking at the different font shapes clearly highlights the diversity of the data set. It is particularly visible with the G-clef shapes, as shown in Table 1, with 18 different G-clef shapes.



















 Amadino	 Antico	 Ballard
 Donangeli	 Ang. Gardano	 Gerlach
 Hucher	 Moderne	 Petrejus
 Phalèse	 Scotto	 Short
 Susato	 Vissenaken	 Waelrant
 [unknown]	 [unknown]	 [unknown]

Table 1: The font and woodcut shapes in the books used for the evaluation are highly variable as illustrated by the 18 different G-clef shapes appearing in the data set.

¹⁸ J. Bernstein: *Print Culture and Music in Sixteenth-Century Venice*, Oxford, Oxford University Press, 2001.

Looking at the EMO collection and evaluating current OMR technology on it provided us with valuable information about the feasibility of large-scale OMR projects. The EMO collection is a good example of what we can expect to find in library collections of similar size. Looking at its content showed that 20% of the sources were not appropriate to the Aruspix application software with its current stage of development. This is an informative figure. However, the EMO collection contains only music anthologies, a subset of the music printing production of the sixteenth century. Looking at the RISM A/I Series, the Series on individual prints, revealed that several categories in the 20% of the EMO collection that were left out would certainly be less represented if we looked at music printing at that time on a larger scale. As an example, in the RISM A/I, only 3% of the sources inventoried are lute tablature, whilst these represent 10% in EMO. Similarly, 0.6% in A/I are prints printed by multiple impression by Petrucci while there are 3% in the EMO collection. This means that quite likely only 5–10% of sources would need to be left out in a larger collection not limited to anthologies.

A baseline recognition rate evaluation was performed on these 260 books of the EMO collection. For each book, one page of music was processed with the Aruspix software application and fully corrected by hand. With this data, it was possible to evaluate how accurately Aruspix performed. Fewer than 1% of the pages were not successfully processed by the software application because of poor image or print quality. The evaluation of the pages that were successfully processed showed that the baseline average recognition rate can be expected to be between 85% and 100% for about three quarters of the books, with a median recognition rate of about 90%. The results of these experiments are presented in more detail by Pugin and Crawford.¹⁹ It is important to note that this baseline was obtained without benefiting from the adaptive feature of the Aruspix software application. It has been shown that in a digitization workflow that would involve some manual correction, even for only a limited number of pages, the accuracy is expected to increase rapidly with the adaptive feature.²⁰

From the Parts to the Score

Transcribing sources with OMR can make it possible to automatically build large corpora of music data. Of course, the accuracy of the transcription limits what can be envisaged with the data. For text, it has been shown that 60% of accuracy is enough for making a transcription useful to the user via full-text searching.²¹ Our recognition evaluation on the EMO collection clearly shows that the current OMR technology has a great potential in corpus building. The transcriptions it can produce are quite likely to be accurate enough to be useful to the user, even if searching text is different from searching music and we might need higher accuracy for music transcription to be usable. How useful automatically generated transcriptions can be has still to be evaluated and documented. Certain types of transcription errors can be more problematic than others. Typically, a mistranscribed clef will cause all pitches to be wrong. This will not cause problems if the transcription is used for diatonic interval searching. However, it will be unusable for exact pitch searching or for twelve tone interval searching.

In some cases, such problems will not be due to the inaccuracy of the transcription but rather to the sources themselves. An interesting case in the EMO collection is the piece *Nymphes des bois* of Josquin composed for the death of Johannes Ockeghem (K.a.3.7). The piece is printed in black notation, although the coloration has only a symbolic meaning. Another peculiarity is that the piece has no clef, probably to signify a detachment. Even if the clef can be deduced from the two flats of the key signature it is a typical example that illustrates the limits of the automatic transcriptions.

19 L. Pugin, and T. Crawford: "Evaluating OMR on the Early Music Online collection," in *Proceedings of the ISMIR 2013 Conference*, [forthcoming].

20 L. Pugin, J. A. Burgoyne, and I. Fujinaga: "MAP adaptation to improve optical music recognition of early music documents using hidden Markov models," in *Proceedings of the ISMIR 2007 Conference*, pp. 513–16.

21 G. Cron: "Corpus." Presentation, Journée IMPACT, BnF, Paris, September 19, 2011.

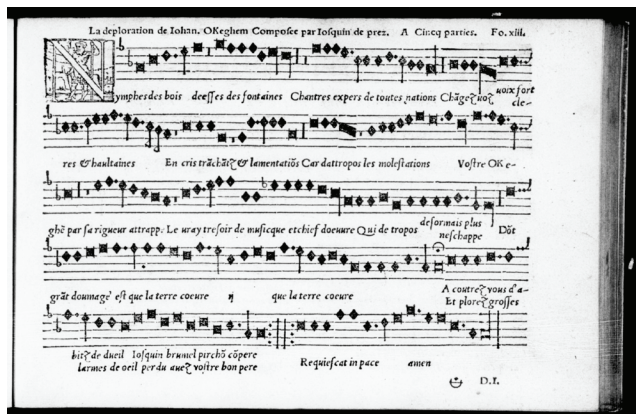


Figure 7: *Nymphes des bois* composed by Josquin printed by Tielman Susato in Antwerpen in 1545. The piece is printed in black notation with no clef, causing additional transcription difficulties.

The Score

One limitation of a corpus of transcribed part-books is that it can be made searchable or can be used for data analysis only at a monophonic level because the parts are given separately. Overcoming this limitation requires the parts to be assembled, which is a challenging task. Two challenges can be identified.

The first is to be able to determine which pages belong to which pieces. In some cases, when the part-books are structured in a very systematic way, the task can be quite straightforward. The part-books were frequently organized systematically because the music printers used to favour it. For example, in order to make the printing process as simple as possible, printers always tried to apply the same layout pattern. This made the typesetting process simpler, with even the possibility of keeping text elements on the printing forme from one page or one part to the other. This can be observed easily from typesetting errors, such as wrong page numbering, or any other anomaly such as broken types that often appear repeatedly on all pages or on all parts in exactly the same place. The best-case scenario for the printer, and for us today when building a corpus, was to have one piece per page, for example one madrigal per page, and also to have the same number of pieces throughout the books. This made the printing process as simple as possible. Each part-book had the same number of pages (or folios), with the same page numbering and also the same table of contents, which they did not have to re-typeset for each part-book. The form could then simply be re-used as is for all the part-books. Such practices can easily be detected from printing anomalies.

The situation quickly becomes more complicated when such a systematic organization of the part-books is absent. Often each part of a piece is not printed on one single page but across several pages. Conversely, one page may contain more than one piece or section of a piece. Typically, with madrigals of various lengths, printers would try to optimize the use of the paper and fill the pages as much as possible. A long madrigal can be printed on more than one page, with a staff or more on the next page, followed by a shorter one that would fill the rest, as shown in Figure 8.



Figure 8: An example with a madrigal ending on the following page, followed by a shorter one that fills the rest. Assembling the parts requires the content of the pages to be assigned appropriately to each piece.

Another complication occurs when the number of voices in the pieces is not the same throughout the book. A good example is a Canzona anthology printed by Alessandro Raverii in 1608 (RISM B/I 1608²⁴). It is in eight part-

books, plus a Basso continuo part. The anthology contains 36 pieces. There are seventeen pieces with four voices, seven with five, eleven with eight and also one piece with sixteen voices. Because there are only eight part-books, the piece with sixteen voices is printed as what can be interpreted as a mixed mode of part-book and choir book, with two voices in each part-book. For this piece, each part book is expected to be shared by two musicians in the same way musicians share a choir book. The assignment of the voice-parts in the part-books of this collection is illustrated in Figure 9.

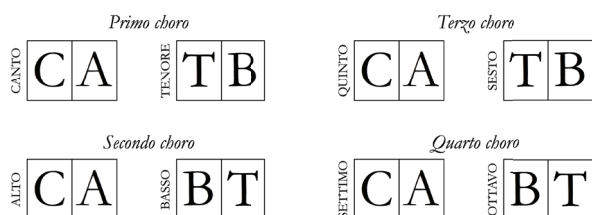


Figure 9: The organization of the voices in the anthology of Raveri printed in 1608. The piece with sixteen voices (four choirs) is printed in the eight part-books with two voice-parts in each part-book.

We can observe patterns in the layout of the printed books that made it easier for the printers to organize the printing process, from typesetting to binding. However, the typesetting layout of music notation was unconstrained, and the printers had the freedom to lay out the music as needed. In Figure 10, we can see one example of a print (K.10.a.9.4) that does not correspond to a common pattern. This shows a piece in three voices. Two voices are printed in choir book format, one per page, and one voice underneath is in *libro aperto* where the music is read from left to right across the two pages.

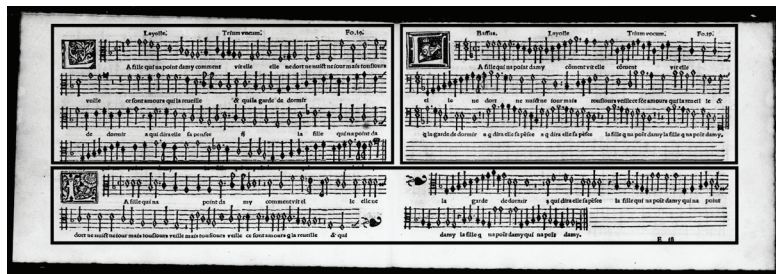


Figure 10: A piece from *Le parangon des chansons* printed by Jacques Moderne in Lyon in 1539. One voice is printed below the two others in *libro aperto* across the two pages.

In the simplest case, with one piece per page and one page per piece, with the same number of voices throughout the book, grouping the parts to build the score is fairly trivial (given perfect note-recognition). With the EMO collection, there is data on the book structure that can be directly useful for this. Indeed, for each book the EMO collection also includes information about the content of each image, and more precisely to which part it belongs, as shown in Figure 11. With more complex cases, it is possible to envisage approaches using intelligent text-mining based on text elements of the part-books extracted via OCR, such as voice labels (Soprano, Alto, etc.), lyrics, or piece titles. However, any of these solutions will certainly require human supervision and verification.

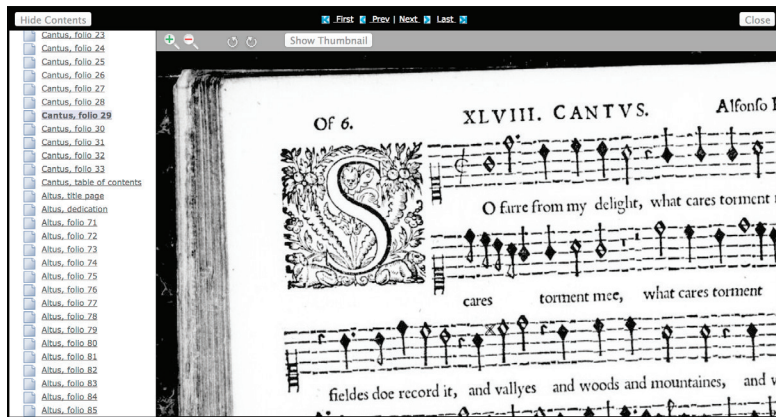


Figure 11: The browsing interface in the EMO collection. For each book, the content of each image is given with information about the part.

The Durations

Another challenge when assembling part-books is the alignment of the voices in ternary meter. In ternary music, the durations in mensural notation are given by their context. This means that recognizing a symbol (breve, minim) does not provide all the duration information, which is given by the symbol together with the context in which it occurs in a similar way to reading pitches. A pitch is given by the position on the staff line, but not exclusively so. The clefs and the key signature define a context that must be taken into account for interpreting the pitch (and octave) of a note. Reading duration is a similar process. The duration is given by the symbol but also by the current proportion sign. In addition to this, the local context must be taken into account. More precisely, the notes before and after determine how the alteration and imperfection rules apply. The fact that both the notes before and those after must be taken into account makes the process complex. We could say that this would be as if, for pitches, subsequent alterations had to be taken into account, or if the pitch of the notes in a chord were determined by looking at notes below and above it. In other words, a purely sequential decoding is not possible.

However, because the system is essentially rule-based, one might imagine the problem could be solved easily by a computer program. This is probably true to some extent. We can imagine solving the problem by representing the duration in a hierarchical structure to which the different rules would be applied recursively. However, to our knowledge, the problem has never been solved. The only reference we have found on interpreting rhythm in early notation by computer is the work by N. Rubinstein.²² Her conclusion that “even the human musician, confronted with a line of Franconian notation, is sometimes reduced to using a method of elimination and speculation” confirms the difficulty of the task, even when dealing with Franconian notation that has fewer levels of durations than 16th-century mensural notation.

The difficulty is certainly due to the fact that the problem is not limited to the application of the rules. Most of the time, transcribing the duration requires ambiguities to be solved. There are indeed many examples of ambiguity in the ways signs were used. Typical examples are how the modes were notated (e.g., with silence signs), that no mode indication should be understood as ternary or binary depending on the region, or that proportion signs or colorations were used with different meaning. In other words, transcribing the notation requires the historical context to be taken into account. As a result, the best approach for aligning the parts is probably to have mixed and flexible methods guided, as for assembling the parts, by human supervision and verification.

22 N. O. Rubenstein: “A FORTRAN computer program for transcribing Franconian rhythm,” PhD Dissertation, Washington University, St. Louis, 1972.

The Encoding

In corpus building, the encoding scheme is a fundamental question. It has some direct implications for current and also future interoperability. The encoding scheme determines if and how the data will be usable by tools and how it can be shared with other projects. The encoding scheme is also directly related to the long-term archiving question. With early music collections having heterogeneous book formats and notation types, adopting an encoding scheme that can be common to each or at least several of the formats and notation types appears to be essential. Having a common scheme makes it possible to share data between projects dealing with different notation types, including metadata that very often have the same components. To some extent, sharing a common encoding scheme also makes it possible to share tools.

Considering these points, MEI is clearly the best candidate for encoding part-books. MEI has the flexibility for modeling various types of music notation, including mensural notation. It includes a dedicated module specifically designed for accommodating the special needs of mensural notation. MEI also includes a rich metadata header that makes it possible to include very detailed metadata in a highly structured manner. It has the advantage of being community-driven and scholarly. Finally, MEI has by default all the benefits of any XML format. Standard tools for manipulating XML can be used, and it is machine-readable and at the same time human-readable, at least to some extent.

MEI also provides an encoding organization by parts (as opposed to a score organization), which makes it perfectly suitable for encoding part-books. In a part organization, each part is encoded in a distinct XML sub-tree. With this, it is possible to have all the parts in one single file even if the score has not yet been assembled. Another advantage of MEI and its flexibility is that it makes it possible to encode duration both in the visible domain (the sign as represented on the source) and in the logical domain (e.g., its actual duration). This is perfectly adequate for encoding a transcription of interpreted duration in ternary notation.

In corpus building, however, the question of how to switch from a part encoding to a score encoding is not an easy one. What link to the original source do we want keep (or can we keep) once we have switched to a score representation? Can we create a diplomatic score transcription of the part-books? The question is directly related to the distinction that must be made between the source and the work as represented in the source. With part-books, the representation of the work still requires the parts to be assembled. This used to happen during the performance and was left to the musicians. In a corpus, we need to encode this representation. Even if we aim at building a diplomatic score transcription, encoding the score creates a distance from the sources in part-books since the organization is no longer the same. As an example, keeping ligatures in a score is cumbersome, and scholars traditionally use artifices such as brackets to mark them in the score.

A possible solution to this problem was proposed in experiments for introducing a layout module in MEI. The general idea of the layout module is to separate more effectively the presentation and the content.²³ With the layout module, there is one single XML sub-tree for the content (the usual MEI sub-tree with the actual music encoding) and in addition distinct sub-trees for each of its presentations. The elements in the presentation (or layout) sub-tree always point to the content. With the part-books, the parts are one presentation of the content and the score is another. MEI with this experimental layout module is currently being used in Aruspix with promising results. However, while this approach clearly shows advantages by avoiding data redundancy, future work on this will be necessary to determine if this is a valid option for encoding large corpora of part-books.

23 L. Pugin, J. Kepper, P. Roland, M. Hartwig, and A. Hankinson: "Separating presentation and content in MEI," in *Proceedings of the ISMIR 2012 Conference*, pp. 505–10.

Vladimir Viro, *Processing Millions of Pages of Music: A Case Study*

Building a large corpus of musical data and making it accessible and useful for a large number of diverse users is a challenging task. Due to the large amount of data to be processed the system has to be distributed and fault-tolerant. Since the majority of the data comes from diverse sources we need to support various input formats. The large number of users means that the front end needs high availability, security and scalability. The diverse user base that includes professional musicologists and a wide array of applications that can be built upon the data sets we are assembling requires flexibility and extensibility of the underlying data storage and processing layers. It also requires an easy way to provide APIs upon which to build user-facing applications. As a research project we need to make the system as cost-effective as possible on top of all other requirements. In this overview we share some operational details of our system, the solutions we chose to overcome certain difficulties, and present some applications that become viable when a large and diverse database of imperfect digital score material becomes available.

We begin with an overview of the overall architecture, of our data ingestion and processing pipeline, go on to describe the system's backend and frontend, and close with an overview of a few applications built on top of the presented components.

Architecture Overview

Our systems consists of three main layers: the permanently-running VMWare server that hosts the main database and where the ad-hoc transformations are applied to the data, the frontend Peachnote.com hosted on Google App Engine, with which the end users of Peachnote applications interact, and a number of virtual machines running on Amazon's EC2 that are used on demand for data ingestion and processing. For the database we use HBase, a schema-less key-value storage engine running on top of Hadoop framework that supports MapReduce which we use for various data transformations. We will describe the backend in more detail in section 3, the frontend in section 4, and now focus on how we collect the data in the first place.

Data Ingestion Pipeline

Obtaining and processing hundreds of thousands of scanned music scores using only one computer would take a very long time. Doing so in one run would most probably not work due to intermittent errors and would be impractical because improvements in handling certain processing steps would require reprocessing of all other steps as well. For these reasons we have designed our system in a distributed fashion in which the components are decoupled from each other and can be run asynchronously and in parallel.

Inter-Process Communication

We handle inter-process communication using the Amazon Simple Queue Service (SQS) – a fast, reliable, scalable, fully managed queue service. SQS makes it simple and cost-effective to decouple the components of a distributed application. One can use SQS to transmit any volume of data, at any level of throughput, without losing messages or requiring other services to be always available.

We serialize our messages using the JSON format for easy inspection, creation and consumption. For example, an OMR task could have following simplified representation:

```
{ "scoreId": "IMSLP00001", "url": "http://erato.uvt.nl/files/imglnks/usimg/8/8e/IMSLP00001-Beethoven__L.v._-_Piano_Sonata_01.pdf" }
```

The processes poll the job queues, fetch tasks that for a defined period of time become unavailable to other workers, process the tasks, and on completion or determination that no other worker should process the same task again remove the task from the queue. The queues can hold tasks for up to 14 days. The workers check whether their output queues contain a number of tasks higher than a certain threshold, in which case they stop processing and wait for the downstream workers to process the accumulated tasks before sending

more jobs. This way we balance the workload and avoid building up jams in the pipeline. This behaviour is of course optional and can be regulated depending on the processing resources available.

Crawling

We draw the bulk of our data from a few major sources of musical data: the Petrucci Music Library, certain collections of the Library of Congress, and the Internet at large.

Petrucci Music Library

The Petrucci Music Library (IMSLP) is the largest online library of public domain sheet music with over 250,000 scores of over 71,000 works. With an average score on the IMSLP containing 13 sheets, it holds over 3.2 million pages of music. Most of them are scanned, the minority are however digitally born. The quality of the scans varies a lot. More than 95% of the material is available in the PDF format. The library is maintained by volunteers, but the quality and depth of the metadata associated with the scores and works is surprisingly high. The site runs on Mediawiki software. The combined size of the PDF files is over 450 gigabytes. The collection is growing steadily by about 1,000 scores a week on average.

In order to obtain the metadata we crawl the IMSLP website regularly. We use Amazon EC2 for distributing the crawling and speeding it up. With 50 machines the full crawl takes about 8 hours (every machine throttles the crawling and fetches only 3 pages per minute). Using the Mediawiki API we are able to crawl only the subset of the pages that have been recently updated. We store the complete content of the web pages in our HBase backend for later processing (more details in section 3).

The scores themselves are fetched by the processes that split the PDFs into separate images of single note sheets (more details in section 2.3). These images are then stored on Amazon S3 for later processing and display. We do not store the original PDF files.

Library of Congress

The Library of Congress hosts the *Music for the Nation: American Sheet Music* collection containing more than 62,500 pieces of historical sheet music registered for copyright between 1820–1860 and 1870–1885. The images are released in public domain. The scores are stored page-wise with high resolution scans available in TIFF format. Since the collection is not growing we needed to crawl it only once. Writing the metadata extractor and crawler took us a few days. Since we could crawl the materials without limiting the rate, the crawling was accomplished using only one machine.

Internet

Going beyond particular sheet music collections such as IMSLP or Library of Congress we are interested in crawling music from all over the Internet. The intention behind this is not necessarily obtaining new or high quality material. Rather, it is providing more context for already known pieces. People often post images of score excerpts on the web when they are discussing and analyzing the musical works themselves. If we can find and link such images with the rest of the well annotated sheet music from library collections, we can provide users with background and discussions of these works that are available online.

We are relying on Bing Image Search API for the initial set of potentially relevant images. We use the Analytics data from the Petrucci Music Library to assemble a list of about 100,000 search requests related to sheet music. We send these requests to the Bing Search API and use the results to retrieve the images, which we then funnel into our standard OMR and indexing pipeline. The image crawling is accomplished using hundreds of virtual machines rented on Amazon EC2. This way in the course of about 24 hours we have collected over 4,200,000 images, of which about 25% actually contain notated music. With more funding the crawling could be easily scaled by an order of magnitude or more. The already indexed images are available via our search engine Peachnote.com

Using the web as a data source has its own issues, most visible of which being a relatively short lifespan of an average image or web page²⁴. Either the whole pages should be archived, or the crawler should run constantly in order to guarantee the freshness of the data.

24 The Internet Archive claims an average lifespan of a web page being 44 –75 days.

PDF to Image Conversion

Our largest source of high-quality, well-annotated data is the Petrucci Music Library. The library carries the scores mostly in the PDF format. The files are submitted by volunteers and originate from a large variety of sources, ranging from large-scale library digitization efforts to scans and digitally notated scores by amateur musicians. Thus the PDF files vary considerably in their properties, which is helped by the richness of the PDF format. Since scores can be quite large and the OMR software is not at all robust (outright crashing on 12% of the sheet images from the IMSLP collection), in order to achieve the highest overall recognition rate we need to process every image separately, eliminating the chance that one image, that itself would bring the OMR software to a halt, would cause other images from the same score to not be recognized as well. In order to recognize the images separately we need to split the original PDFs into separate sheet images first.

Surprisingly, the task of converting PDF files to images is not a straightforward and easy one. Various open-source and commercial software libraries and components exist that have this procedure as their purpose. Due to PDF being a container format with lots of options for the way the actual content is encoded (the images can be encoded using various schemes), many libraries cover only a subset of the PDFs we are interested in. Ghostscript is one of the best available options for converting PDFs to images, but one has to take care before applying it to the data. The reason behind this is the vector nature of some scores.

Ghostscript needs to know the resolution of the image to export to. In case of scanned images it takes their resolution by default, and this is what we want. In case of vector-based images there is no inherent notion of resolution and the software falls back either on some defaults that do not make sense for the score at hand, or uses some embedded image that it could find in the vector file. A considerable number of vector images have a white background image of a small resolution, e.g. 595 x 842, and this is the size that is detected by Ghostscript using the MediaBox scaling heuristic. The resulting image is too low-resolution for OMR to process successfully. So we reprocess the images with low implied resolution and render them in a higher resolution of 300dpi in the hope that the original was a vector file and we do not just end up with an upsampled version of the same raster image. This heuristic could easily be improved by determining whether the PDF file is just a flat scan as opposed to hybrid or completely digitally born image by examining the PDF content programmatically.

Image size optimization is also an important topic since we are dealing with millions of sheet images with total size exceeding several terabytes. We store the images in the PNG format because it is lossless, enjoys a wide library support and can be displayed by any browser without need for add-ons. It turns out that the same image data can be stored with different efficiencies within a PNG – more efficient representations can reach half the size of less efficient ones. Therefore an additional “compression” step can be useful for the output of the PDF splitting process. We use the pngquant library for this purpose.

OMR

While we are developing our own OMR algorithms that should remove our dependence on closed-source software and provide a better recognition quality, we are currently using Smartscore and Photoscore, the best commercially available OMR systems, for the bulk of our OMR needs. The systems are not available as software libraries and can be used only through a GUI. In order to employ them in our automated pipeline we have automated the process of using them through their GUI. The software runs in virtual machines, obtains its tasks from the SQS queue, fetches the images from specified locations, processes them, and writes the results to the S3, while also writing the metadata and logs to a distributed and reliable database hosted by Amazon (DynamoDB). This way the whole OMR component is reliant only on the Amazon-provided services (SQS, S3 and EC2), which is good reliability- and scalability-wise.

Since the OMR software is not reliable (it crashes and hangs on a number of provided images) we need to detect and kill hanging processes in order to continue the recognition. Since we are dealing with closed-source software, we need to apply heuristics in order to determine when to stop the processing and move on to the next image in order to improve the throughput of our pipeline. We are doing this by automatically observing the progress indicators and trying to guess as early as possible whether the OMR process has stalled. Apart from such heuristics we have a hard timeout of two minutes per image, which when triggered kills the OMR process. We of course store the run-time information of the OMR process in a database, so that if we later

revise our heuristics or decide to give certain failed images a second chance to get successfully recognized, we know at which stage the OMR process has been killed last time.

Periodically we collect the OMR results from the S3 storage, import them into our main database, index them and make them searchable and usable by our applications.

Backend and Data Explorability

For our backend we use the Hadoop framework and the schema-less HBase columnar database on top of it, which is scalable and makes exploratory data mining based on MapReduce convenient. One of the very important features of HBase is its support for transparent compression, which allows us to store a very large inverted search index of over a billion n-grams rather compactly.

Columnar Storage and MapReduce

The schema-less nature of columnar HBase storage and its integration with MapReduce facilities of Hadoop make it easy to prepare and transform the data incrementally without the need to design the schema upfront. New index types can be added easily – we only need to add a function that splits a MIDI or MusicXML file into a list of n-grams that we want to index it by. We can plug in this function into the system and run a MapReduce job for creating the search index. Once the results are available, the index can be accessed via a REST API and researchers can start using it immediately. This extensibility makes it easy to provide new views of the data and build applications on top of them. An interesting direction of future research would be to expose this plug-in functionality to researchers over the internet so that they could define new interesting views of the data completely by themselves.

System Administration Issues

It should be noted that it is better to run Hadoop and HBase on a good hardware configuration with enough RAM and fast networking between Hadoop nodes. Otherwise the time needed for administration might at some point start outweighing the benefits that the framework provides.

Currently the infrastructure is running inside the virtual machines. This makes it easier to migrate or clone it to new hardware. Previously this was not the case and the migration of the main server took about two weeks of system administration work, as well as some time for re-building the indexes.

Frontend

Our applications are used by thousands of people every day and thus security, availability and scalability are important to us. We lack expertise in running web services securely and reliably, so we outsource this part to our platform-as-a-service provider of choice – Google. Our users interact with the services exclusively via App Engine applications that in turn communicate with our main database and serve as a caching layer. App Engine makes updating applications easy by providing an easy deployment path and an ability to switch between multiple application versions.

App Engine offers an uptime of over 99.5% and handles a steady load on our applications of 12–30 requests per second throughout the day, with occasional spikes on some of them of 5x–10x of the normal load. Both our end-user applications and the API are running on the App Engine. The Score Search and Score Viewer applications are developed in Java using Google Web Toolkit, but for more exploratory work for which Java would be too heavy-weight we just expose the data via an API²⁵ (with JSONP support for cross-site third-party use) and build the interface directly in JavaScript.

²⁵ Documentation available at <http://www.peachnote.com/api.html>

Applications

Score Search

Score search is the conceptually simplest application of OMR at scale. But enabling a full text search for a large collection of music can go beyond simple occurrence retrieval. When combined with score metadata, search can provide insights into musical history. The best known concept of this kind is the n-gram viewer popularized by Google.^{26,27} We have applied the same concept to music and released the musical n-gram viewer.^{28,29}

Score Similarity

An interesting question to ask when dealing with lots of scores is can we provide a good measure of similarity for scores that we have collected? One practical application of such a measure would be identifying scores and MIDI files based exclusively on their content (since the metadata are not always normalized or even available, as is the case for a lot of MIDI files). The more correspondences we can find, the better use we can make of the redundant data, for example for cleaning up OMR mistakes. Also, it would be interesting to see whether the similarity measure could go beyond duplicate detection and could help us organize the scores meaningfully.

We measure similarity between scores based purely on their content, i.e. the music itself. No metadata are used. The algorithm relies on a huge inverted index used in our score search engine, Peachnote.com. The algorithm is pretty straightforward and works as follows. We scan through the inverted index (the mapping between musical n-grams, in our case chord progressions, and lists of scores they can be found in). We consider only rare patterns (n-grams that occur at most 40 times in our whole data set of over a million score sheets). For every such rare pattern, for every pair of scores from the list of scores containing it, we increment a co-occurrence counter of the score pair by one. Then, for each score we store a list of scores such that the accumulated count for the pair containing both scores exceed 20 (we store counts as well). This way, for every score containing rare musical patterns we obtain a list of other scores that share at least 20 rare patterns with that score. This is the complete description of the algorithm. In short the main idea is following:

The rarer (and thus more characteristic) patterns a pair of scores has in common, the more similar we judge them.

We find it surprising that this simple definition of similarity seems to capture musical semantics so well. Not only does this similarity capture the specificity of a particular piece; scores by the same composer, of the same genre or the same time are rated more similar to each other too.

The algorithm is implemented in a single MapReduce job running in our Hadoop environment. On our setup the algorithm runs in about 40 minutes. We use the chord n-gram type. Rhythmic information is not considered.

The algorithm could be improved by using normalization (as a second MapReduce step). This way we could improve the quality of ranking and avoid the problem of hubs (large scores sharing many rare patterns with lots of other scores). Examples of such hubs currently are some large scores by Bach, orchestral scores of Fidelio and some Wagner operas. We plan to implement the normalization in the near future.

Interested readers can explore the algorithm's results online³⁰ (see Figure 12).

26 See <http://www.peachnote.com>

27 Jean-Baptiste Michel*, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K. Gray, William Brockman, The Google Books Team, Joseph P. Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker, Martin A. Nowak, and Erez Lieberman Aiden*. Quantitative Analysis of Culture Using Millions of Digitized Books. *Science* (Published online ahead of print: 12/16/2010)

28 See <http://books.google.com/ngrams>

29 V. Viro: "Peachnote: Music Score Search and Analysis Platform," *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, 2011, pp. 359–362.

30 Available at <http://www.peachnote.com/similarity>.

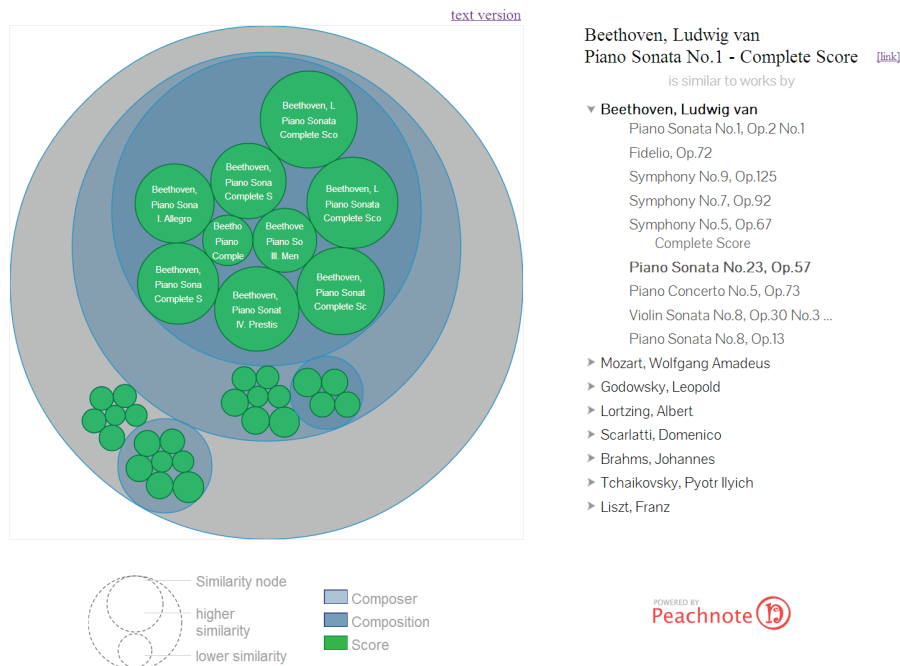


Figure 12: An interactive interface of the similarity browser. The scores are grouped by composition and composer.

Audio Synchronization

A large data set of symbolic music data can be enriched with sensory data. By synchronizing audio recordings with sheet music we can derive a tempo commonly used at certain positions in a piece. We can also provide users with a quick way of finding out how others perform a certain place in a musical piece which they are interested. The users can just click on a place in the score and see different recordings synchronized with the score right at the selected position.

Interested readers can explore the algorithm’s results online³¹. We plan to release this feature on the IMSLP website in the near future (see Figure 13).

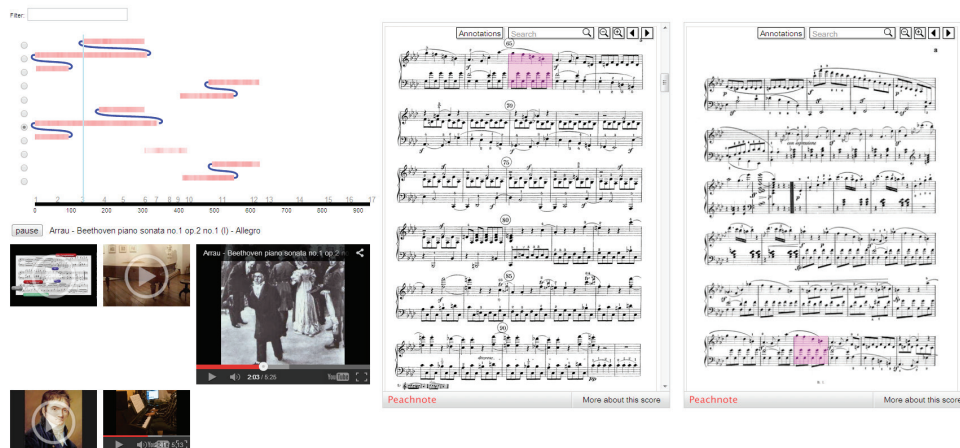


Figure 13: The alignment browser. The scores from the Petrucci Music Library are automatically synchronized among themselves and with multiple recordings from YouTube. The graph represents piecewise alignments between the videos and the score. The x-axis represents the score time. The colours of the aligned video segments represent the deviation of the performance from the dead-pan tempo. For displaying and interacting with sheet music we use our Score Viewer³².

31 Available at <http://www.peachnote.com/sync>.

32 Available at <http://www.peachnote.com/viewer>.

Discussion

The practicalities of corpus building mentioned in our title are illustrated by a set of very different projects, each based on different collections and different expected uses and users. In some cases, the resource must be constructed within the project, in other cases, it is discovered and collected. It is the practical aspects of the activities of building a corpora and making it accessible and searchable that give such projects their shape. These practicalities may come in the form of limitations and constraints, but they may involve user needs only revealed by real-world deployment, and provide unexpected opportunities and widened perspectives.

In the various projects discussed here, differences of approach can be seen as originating in these practicalities, in the diverse nature of the materials, scale and expected users of each.

Materials

Producing a complete edition of Tinctoris treatises requires grappling with a relatively small number of manuscripts, but involves the close examination of different hands and layouts, and mixed text, images and several musical notations. ECOLM and EMO have been handling larger volumes of sources, but ones which are the products of a more constrained and consistent period of music typesetting. In the former case, manual entry is necessary, in the latter, OMR has now been demonstrated to be a practical first step, albeit one that has an influence on subsequent editing and dissemination.

A more complex situation is to be found in IMSLP, where a large digital corpus already exists or is gathered from sources on the Internet, but where the resource is extremely heterogeneous. The size and variable quality of the data makes statistical, approximate and incomplete approaches a necessity, and dictates the sorts of questions that can reasonably be asked of the resources. What is lost in fine detail may be compensated for by the sheer scale of music available to those uses that are robust to that loss.

Diversity of materials and approaches is a desirable aspect of both humanities scholarship and arts practice. It should not necessarily be regarded as a problem to 'solve' in the negative sense, and handling the range is part of the skill set required by the discipline. MEI is a format that has expanded outwards from a core of standard classical music notational practice. The breadth of notation and encoding requirements seen in this discussion, though, must surely constitute one of the limits on the current use of MEI, and is one reason why none of these projects – each, in its own way an edge case – uses the format natively internally for storage or processing.

Users

A corpus is not just a silo for storing information – it is fundamentally a resource for users. Who those users are and what their needs are is an important, if often unspoken, factor in corpus building.

The Tinctoris edition, ECOLM and the projects that use Aruspix can generally be characterised as musicological and positivist. Although there is an expectation that much of that use will come from outside of academia, the musicians who consult these collections are as likely to expect fine detail and precision as professional academics. The projects have generally required very specialised information to be encoded which, perhaps, has resulted in MEI being given second-class status – delaying support for it, using it purely as an export format or largely ignoring it. Where Aruspix now supports export into the format, it is because MEI has been extended to embrace the richness of representation achieved already in the native format and internal representation of Aruspix: Pugin must be thanked for adding many vital elements to the specification.

Users of IMSLP have different demands and, although some will overlap with those for more musicological projects, the vast majority use the corpus as a resource providing sheet music, and expect to find discovery tools for the purpose centred primarily around metadata. Once this functionality had been put in place, further tools, such as those for content-based retrieval and for exploring the nature of the corpus itself could be added, catering for a wider range of users.

There is a tendency in MIR, but especially in corpus building, to consider user needs as secondary. On the surface, the most important thing to achieve first is to get 'the data' or 'the music' into a form which is machine readable and then make it human accessible. It is for that reason that the editors have been a significant focus for software development for the Tinctoris edition. Further consideration should make clear, however, that 'the music' is decided as much by the receiver as by the source. There are increasing numbers of studies of

user practices, for instance by Barthes and Dixon³³, but as yet few signs of these making their way into project designs or, perhaps more interestingly, of projects with explicit user studies built in to the early stages of their work plans.

Nonetheless, several of the projects discussed here are committed to actively involving users in aspects of their resources. We have seen various ways of engaging a community here, and many others are used elsewhere. In ECOLM, the effort has, so far, been to harness the free time of capable amateurs to reduce the expense and effort of the editorial process. For the Tinctoris edition, a fairly conventional use of social media and a blog is intended to give the developers information about user practices and intentions, in the hope of better identifying and catering for user needs. The extent to which their responses will feed back into the project's development and design remains to be seen – there will be less than a year between the first comments on the blog and the end of the project's funding. There is virtue in a 'top down' approach, followed by consultation, but there are also other models, and these remain, as yet, largely unexplored by the musicological community.

The relationship between project and user – whether totally passive or, at least partly, exploitative – varies hugely. The risks of crowd sourcing have been widely discussed elsewhere, and those who wish to harness this invaluable resource must develop appropriate strategies: strategies for motivating users, whether through gamification or credit on a leader board or some more domain-specific means; strategies for ensuring the accuracy and reliability of contributions; and strategies for recruitment. Clearly any involvement of the users in the encoding or editing process requires both robust software and a relatively easy, light encoding load. For ECOLM, that has meant an entirely different editing paradigm, specialised for correcting rather than entering texts, and designed with ease of use as a primary criterion.

Resources

Resource limitations are characteristic of all projects, and especially interdisciplinary ones. The need to develop, over the course of a few years, an infrastructure that will be sustainable subsequently with no further technical input, and often only limited maintenance of any sort, can have an important impact on the development process. The effect of this limitation in funded projects warrants detailed study, but it seems reasonable to suggest that it may limit risk-taking and large-scale software architecture design, favouring local, beta-level development focussed on relatively short-term goals.

Research almost inevitably generates and uses code that is not of industrial quality or reliability, but software written under significant resource constraints may have limited potential reuse and even be hobbled from further development. Resource restrictions can also encourage local solutions to software and encoding issues, if the adoption of external approaches would carry a greater initial resource cost or an uncertain return on resource investment. This is likely to be another source of slower uptake of MEI and similar technologies if the estimates made by project instigators suggest a relatively low return on resource investment within the lifespan of their project, or a larger level of risk or uncertainty.

The problem of sustainability in research software has long been well known, and initiatives such as the Sustainable Software Institute, along with much of the work carried out under the aegis of MEI, can be seen as attempts to ameliorate the problem. 'Quick and dirty' approaches clearly carry benefits, but there are numerous well-documented strategies within software engineering and project management that mean that such approaches need not be incompatible with code and data sharing and the incremental gain of software libraries by the community as a whole.

MEI in Practice

So, what are the implications of these factors for MEI? It would be unreasonable, and against its design, to expect it to be perfectly suited to all data and all applications at all levels. Similarly, there is no reason to expect it to become ubiquitous overnight.

Rather than specifying a list of required actions and missing features for the developers, it is likely to be more helpful to consider what will be offered to the MEI user at that future point when the practical benefits for project instigators are more apparent. These attributes are most likely to arise partly as a result of planning,

33 Mathieu Barthes & Simon Dixon (2011), 'Observations of Musicologists at the British Library: Implications for Music Information Retrieval' in *Proceedings of the 2011 ISMIR 2011 Conference* (Miami, 2011), pp. 353–358

but also as a result of the work of pioneers extending the state of the art for their own purposes and their own corpora.

MEI still appears to be in its pioneering period, in which most projects need to devise new software and extensions for their own use. Such a period must surely be finite, and the resources in this session and that on chant between them account for a large amount of European notated music. Whilst individual details may be expected to change, it seems reasonable to expect the degree of extension required by new undertakings to reduce over the coming years. Information that is not immediately notational is also part of the 'materials' of a corpus, and, for instance, the extension Pugin discusses to encode graphical layout information is a further step towards an out-of-the-box readiness of the format.

Mechanisms for entry and editing are also increasingly available, but the vast differences in types of musical material and in the process and character of input activity make this a difficult area. IMSLP's use of MusicXML is partly dictated by the commercial OMR software that is used. The success of MusicXML with industrial partners is the product of an entirely different strategy – and product – from that of MEI. To a large extent that side of the problem has been solved admirably by MusicXML, and the presence of a single format, despite – and probably because of – its limitations is helpful, for as long as conversion is possible.

Similarly, clear and practical design patterns, following from use case could make work of the sort described here easier, cheaper and less risky, encouraging more uptake, and more development.