

Towards a Workbench for Symbolic Music Information Retrieval

David Bainbridge
Department of Computer Science
University of Waikato
Hamilton, New Zealand
+64 7 838 4407

d.bainbridge@cs.waikato.ac.nz

ABSTRACT

This paper describes work in progress on a workbench for symbolic music information retrieval (MIR). Drawn from three broad techniques used to perform symbolic retrieval—state-based matching, dynamic programming, and text IR—the workbench implements several variations based on these techniques. To perform an experiment, the user can either interact with the workbench through command line options, or develop scripts that batch process the commands. The latter is preferable when running a series of experiments. The workbench is also designed so it can be embedded into a digital music library to provide content-based querying. Development of a graphical user interface is planned for the near future.

1. INTRODUCTION

This white paper describes work in progress on a workbench for symbolic music information retrieval at the University of Waikato, New Zealand. It is hoped that on completion and its subsequent release into the public domain under the GNU public license, the tool will help research groups compare and contrast their work more readily. The work is complementary to on-going efforts to form a corpus of music for an international community of researchers to use in experiments.

The three main scenarios envisaged for the workbench are:

Use prior to the formation of a corpus for music information retrieval. Unlike the copyright restrictions typically placed on music, the open source nature of the workbench means it is free to be distributed to interested parties. This means the workbench can go to where the datasets are, rather than the other way around, providing a convenient way for different research groups to run the same experiments and forms of evaluation, albeit on potentially different datasets.

Use with an established corpus for music information retrieval. Once a corpus for music information retrieval is formed, a natural extension for the workbench is for it to be used in conjunction with this. For example, when used in tandem this will enable researchers to evaluate more consistently algorithms by performing the same experiments on the same data set. Alternatively, should a new experiment be devised to reveal a particular aspect of music information retrieval the workbench can be used to augment previously published results. The instigator of the new experiment would use the workbench to test the same set of algorithms and dataset used in the previous reported results. This type of activity further promotes a community of researchers working together to

discover the strengths and weaknesses of various algorithms.

Use within a digital music library. Query by content is a valuable feature in a digital music library, and music information retrieval research provides a foundation upon which to build such a service. It is therefore desirable that the workbench be flexible enough that it can be embedded into such an environment. More specifically, there should be the ability for a digital music library to configure the workbench with particular parameters that take account of practical considerations such as the computational complexity of the search, memory usage and recall and precision properties.

Other uses of the workbench, not envisaged by the author, may emerge over time. This is one of the strengths, and the spirit, of open source projects.

Below we detail the software workbench that is being developed, and show how a prototype version is used to power our digital music library system MELDEX. We conclude with some remarks about future developments.

2. A WORKBENCH FOR MIR

Figure 1 gives an overview of the workbench, which was influenced by the group's experience with digital libraries. Written in Java, it is also possible to embed algorithms implemented in other programming languages, such as C ++ , through Java's Native Interface (JNI).

The workbench is divided into two phases: assimilation and runtime. Controlled by a configuration file, the assimilation phase is responsible for collating files together to form indexes and/or databases. The runtime phase is where experimentation is carried out, guided by user input. While assimilation is typically performed once for a given set of experiments, the runtime phase is executed many times to garnish and gather results.

Importing takes care of the multitude of different file formats associated with music data, converting input files into a canonical format that is both general and expressive enough for our needs, yet straightforward to parse. After considering a range of possible formats we selected the XML version of Guido [1]. For each format handled by the workbench, a new object is inherited from the Import base class, and its member functions overridden as appropriate. A mixture of hand-tailored code and, when available, existing open source utilities are used to perform this. For example, the MIDI import module is based around the MIDI to Guido conversion utility (available at <http://www.salieri.org/guido>).

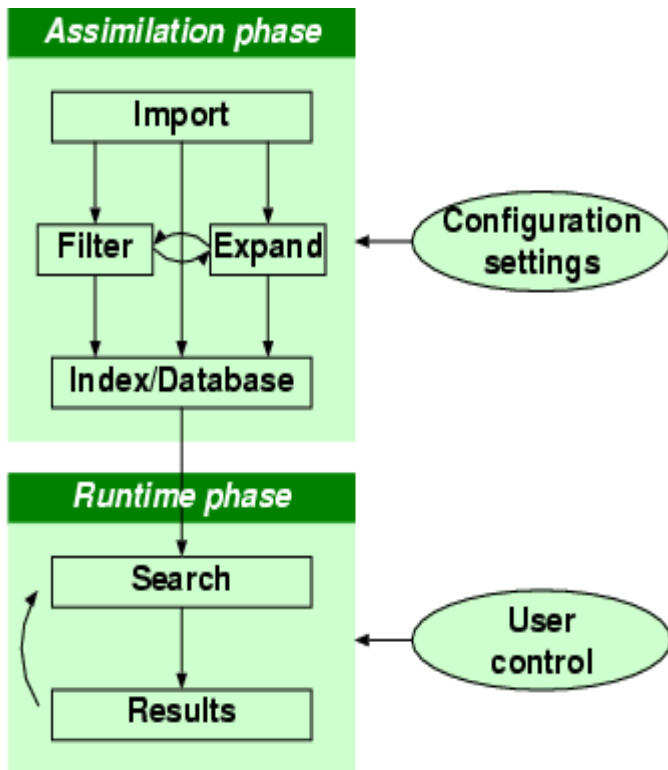


Figure 1: A workbench for symbolic music information retrieval

Filtering and expanding are optional steps. Conceptually the former reduces the stream of musical data passing through it while the latter increases it. From an implementation standpoint, however, this distinction is insignificant and both categories inherit from the same base class that is set up to manipulate (add, remove, or modify) a stream of musical data. The two entities are represented separately in the workbench because it is useful to be able to differentiate between them during the configuration phase. An example of filtering is retaining only MIDI tracks that are monophonic—useful when the searching algorithm being trialed only works on this type of notation. An example of expanding is generating a tune in different keys—useful for algorithms that only work in an absolute key.

A more interesting example of filtering is RepairMotif which detects repeated fragments of tune. Central to the module is the “re-pair” algorithm [2] from which the filter takes its name. This is a rule-based character comparison algorithm that, if coded carefully, detects repeated substrings in linear time and space. To work in a music context, a pre-processing step within RepairMotif maps notes to characters (much like the way notes are mapped to characters in a text IR algorithm used to perform music retrieval), and a post processing step studies the grammar rules produced to pick the most frequent and/or longest substrings and then reverses the mapping for the chosen strings.

To perform searching using the workbench, there is a choice between three types of algorithm: state-based matching [3], dynamic programming [4], and text-based information retrieval [5], each with optional arguments that modify their behaviours—for example matching only at the start rather than at any position within a tune, using a pitch contour rather than exact intervals, and so forth. The output of a search is held as a result set from

which statistics are extracted, graphs plotted and tables generated. The particular operations performed by the workbench in the runtime phase are controlled by the user who can interact directly with the workbench through its command line syntax or develop scripts that batch process the desired experiment.

The group supports open source projects, and there are plans to release the workbench under the GNU public license. It is hoped that this will help foster a stronger sense of community within the music information retrieval field where other groups will add their retrieval algorithms to the workbench, thereby allowing a more comprehensive comparison of the relative strengths and weaknesses of the algorithms being developed.

3. DL SOFTWARE ARCHITECTURE

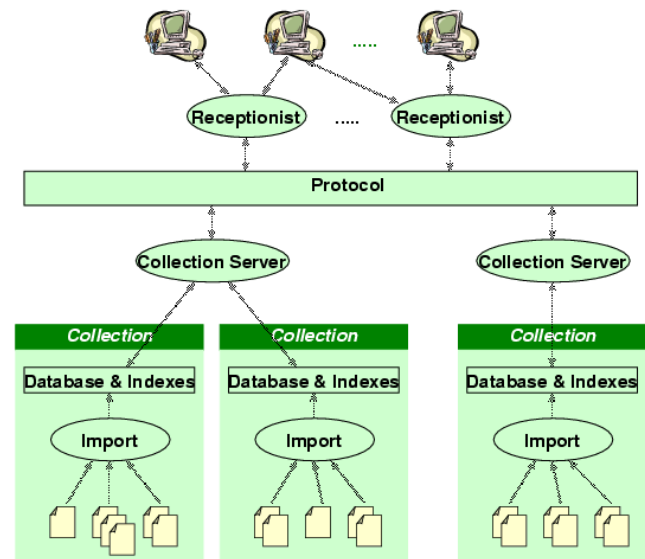


Figure 2: Digital library software architecture

Figure 2 shows the general software architecture devised for the main digital library project [6]. Because documents are represented as abstract entities, the design is flexible enough to handle diverse forms of media, as demonstrated by example collections built from text, images, video and audio documents [7]. To realise a digital music library a prototype version of the workbench was embedded into the general digital library design. This was straightforward as the digital library architecture fore-shadowed the design of the workbench and consequently the workbench was developed with this embedding step in mind.

The workbench assimilation phase corresponds to the lower half of Figure 2 used to build a collection that is searchable and browsable. The term importing in the figure carries a broader meaning in the general design due to the diverse assortment of data types handled. Using a system of “plugins”—modules of code tailored to parse particular formats—source documents in potentially different formats are imported into the collection and transformed into a homogeneous XML format from which the necessary indexes and databases can be formed. The transformation process also includes the ability to augment the data. Examples of this include assigning bibliographical information to audio recordings, and detecting hyperlinks that point outside of the collection (useful when your digital library collection is shipped out as a CD-ROM).

Because importing requirements can differ greatly from one collection to another—even if they use the same data format—a collection has a configuration file that lists the plugins needed and supplies as options to these plugins the specific transformations required. All these abilities tie in well with the music information retrieval workbench: its assimilation phase already incorporates the notion of a configuration file, and the filtering and expanding components drop into place as plugin options.

Less apparent between the diagrams, but equally well aligned in terms of integration, is the workbench's runtime system which maps to the collection server in the digital library design. Although not shown in Figure 2, inside the collection server component there are Search and Result objects, amongst other things. These two objects perform the same function as their workbench counterparts, and through inheritance a new collection music server can be readily constituted that—by accessing the workbench—overrides only the parts specific to supporting music content. The other objects to the collection server remain unchanged, functioning as before providing generic services such as user authentication.

The role of the receptionist component in the digital library system is to provide an interface between the digital library's content, and a user wishing to access it. By decoupling user interface issues from content management a flexible and versatile environment is formed. For example, one receptionist might be setup to be multilingual and interact through HTML pages, whereas another—still accessing the same basic content—might provide a graphical querying environment through a Java client. In addition to this, communication between the receptionist and collection server is through a protocol, enabling distributed configurations of the digital library. In response to a user's query, a receptionist might contact two collection servers running on different computers and present the result, seamlessly merged, to the user. This ability is used in the digital music library to support multiple indexes to large collections that are divided into smaller sections and distributed to multiple computers.

For the purposes of this discussion, our interest is in how the receptionist component fits in with the workbench, and again the answer is smoothly due to forward planning. Whereas in the workbench instructions come directly from the user, in the digital library they arrive via the receptionist. For the workbench to be used to support musical queries therefore, all the receptionist need do is translate these requests into meaningful instruction for the workbench (aided by the specially inherited music collection server), and to format the result into a form the digital library is expecting. The object oriented design of the digital library architecture is such that once more, implementation is a matter of inheriting a new object—this time a specialised music receptionist—that overrides specific member functions.

Used in this way it is true that the full power of the workbench is not utilised by the receptionist, but it is equally true that the receptionist supplies extra services—such as browsing titles A-Z—through other non-workbench related components.

4. CONCLUSIONS

In this paper we have described work in progress on a software workbench for symbolic music information retrieval. Three main scenarios are envisaged: use both prior and in conjunction with a corpus of music for music information; and as the basis for a digital music library.

For experimentation work, the open source nature will encourage the comparison of different algorithms and participation between research groups. Hopefully researchers will be also be motivated enough to add their retrieval algorithms to the system. In fact the terms of the GNU Public License actively encourage this.

Borrowed from the group's existing digital library design—its big brother if you will—the concept of plugins in the workbench provides a flexible approach for importing music files in a range of formats, and the ability for plugins to take options allows for the possibility of filtering or expanding the input stream of music data. Both are desirable abilities given the different characteristics and needs of the datasets.

Looking to the future, while the current emphasis in the group is on symbolic matching the design does not preclude other forms of music information retrieval. Work with raw audio matching, for instance, seems particularly amenable to the approach described here.

5. REFERENCES

- [1] H. Hoos, K. Renz, and M. Gorg. GGUIDO/MIR: An experimental musical information retrieval system based on guido music notation. In J. Stephen Downie and David Bainbridge, editors, *Proceedings of the Second Annual International Symposium on Music Information Retrieval: ISMIR 2001*, pages 41-50, 2001.
- [2] N.J. Larsson and A. Moffat. Offline dictionary-based compression. In *Proceedings of the IEEE Data Compression Conference*, pages 296-305, Snowbird, Utah, 1999.
- [3] S. Wu and U. Manber. Fast text searching allowing errors. *Communications of the ACM*, 35(10):83-91, 1992.
- [4] M. Mongeau and D. Sankoff. Comparison of musical sequences. *Computers and the Humanities*, pages 161-175, 1990.
- [5] I.H. Witten, A. Moffat, and T.C. Bell. *Managing Gigabytes: compressing and indexing documents and images*. Morgan Kaufmann, San Francisco, CA, 1999.
- [6] I.H. Witten, R.J. McNab, J. Jones, M. Apperley, D. Bainbridge, and S.J. Cunningham. Managing complexity in a distributed digital library. *IEEE Computer*, pages 74-79, February 1999.
- [7] D. Bainbridge, G. Buchanan, J. McPherson, S. Jones, A. Mahoui, and I.H. Witten. Greenstone: A platform for distributed digital library applications. In *European Conference of Digital Libraries*, pages 137-148, Darmstadt, Germany, 2001.