PUMP UP THE JAMS: V0.2 AND BEYOND

Brian McFee[1,2,*], Eric J. Humphrey[4], Oriol Nieto[5], Justin Salamon[1,3], Rachel Bittner[1], Jon Forsyth[1], and Juan P. Bello[1]

[1]Music and Audio Research Laboratory, New York University
[2]Center for Data Science, New York University
[3]Center for Urban Science and Progress, New York University
[4]MuseAmi, Inc.
[5]Pandora, Inc.

ABSTRACT

This document describes the changes to the JSON Annotated Music Specification (JAMS) format and implementation between v0.1 and v0.2.

1. INTRODUCTION

The JSON Annotated Music Specification (JAMS) format was proposed by Humphrey *et al.* [1] as a mechanism to serialize structured annotations for musical content. Since the initial publication of the JAMS specification, we (the developers) have learned several lessons in building music information retrieval infrastructure on top of the existing framework. Consequently, we have revised the specification and implementation in various ways to better support a modern and extensible workflow. The purpose of this document is to explain the changes in JAMS following the first publication, describe their underlying motivation, and discuss the potential additions in future versions.

Throughout this document, the previous specification of JAMS as described by Humphrey *et al.* [1] will be referred to as *JAMS-0.1*, while the current specification will be referred to as *JAMS-0.2*.

2. JAMS SPECIFICATION

In this section, we highlight the changes to the JAMS schema definition(s). Since these changes apply to the file structure definition itself, they are independent of the software implementation used to parse or generate JAMS files. [1]

2.1 Unified observation types

JAMS-0.2 reduces all observation types (i.e., observation, range, and time series) to a single format: regardless of task, each *observation* consists of a 4-tuple *(time, duration, value, confidence)*. The *time* and *duration* fields are constrained by the schema to be non-negative numbers. [2] By default, this simulates the *range* type of JAMS-0.1, but taking *duration*$= 0$ recovers the *event* type as well, with a small amount of redundancy.

In order to avoid redundancy and improve efficiency, JAMS-0.2 allows a distinction between *sparse* and *dense* observation lists. Having standardized the observation format, the *Annotation*'s data field, which contains a list of observations, can be interpreted as an $n \times 4$ table, which may be encoded in either a row-major (*sparse*) or column-major (*dense*) format. While the column-major format is generally more spatially efficient, the row-major format is more human-legible, and for most tasks, the difference in efficiency is negligible.

2.2 Task- vs. Annotation-major layout

JAMS-0.2 adopts an annotation-major (rather than task-major) structure. Instead of grouping annotations by task at the top-level as in JAMS-0.1, a JAMS-0.2 object contains a single list of *Annotations*. The annotation-major structure allows for the same core schema to be retained as new tasks are introduced, since there is no explicit dependence on the task definitions. However, since all annotations are collected in a single, anonymous

*Please direct correspondence to brian.mcfee@nyu.edu

[1] The software implementation of JAMS is available at https://github.com/marl/jams.

[2] The *value* and *confidence* fields are left unconstrained at this point, but are defined subsequently depending on the *namespace* as defined in section 2.2.

data structure, we will need a new way to distinguish between annotations for different tasks. This leads us to the task *namespace* abstraction.

2.3 Task namespaces

Each annotation object declares its task through a string-valued field called *namespace*. A *namespace* in JAMS-0.2 is simply a partial schema declaration which defines the following properties:

- an identifier, *e.g.*, "beat" or "tag_cal500";
- schema declarations for the *value* and *confidence* fields;
- whether data should be encoded in *dense* or *sparse* form; and
- a brief plain-text description of the task.

The identifier is included within *Annotation* objects to specify which namespace they should be validated against. The schema declarations for *value* and *confidence* are both optional, but can be used to impose constraints on the permissible contents of an observation. [3]

For convenience, namespaces are grouped into high-level task categories by their identifiers. We stress that this grouping is merely cosmetic, and there is no strict underlying hierarchy of tasks. Finally, namespaces are defined externally to the core schema, and new namespaces can be imported dynamically with no modifications to the JAMS implementation itself. This makes it possible to develop and share custom annotation specifications.

3. FUTURE DIRECTIONS

Many of the existing namespaces are similar enough to share common representations and evaluation schemes, and can therefore be grouped into high-level categories. In some cases, it is even possible to construct explicit mappings between namespaces. This can be useful for simultaneously modeling or comparing data from different corpora. Although complicated, implementing automatic namespace conversion — even if it is occasionally non-invertible — would be valuable for simplifying modeling and evaluation of tasks across different datasets.

JAMS-0.2 provides functionality for local extensions of the supported namespaces, but it can be tedious to add namespace definitions manually in each application. We therefore plan to introduce functionality to support a *local* namespace repository, in addition to the definitions which ship in the main distribution. This repository would be specified by an environment variable or configuration file, and reduce the amount of custom code needed to support local extensions to the namespaces.

Annotations may not span the entire duration of a track. Ideally, annotations should therefore define the time extent over which the annotation is valid. While JAMS-0.2 provided some functionality to encode this (via the *duration* fields or a sandbox entry) it was not standardized, and no provision exists to support partial annotations of zero-duration events. Starting in JAMS-0.2.1, each annotation object will also contain optional `time` and `duration` field. By convention, if these fields are left null, then the annotation should be assumed to span the entire track.

[3] The term *namespace* was chosen to connote that the `value` and *confidence* fields keep the same *name* in different tasks, but their interpretation varies according to *namespace*. This is analogous to the notion of namespace encapsulations in software engineering.

4. REFERENCES

[1] Eric J Humphrey, Justin Salamon, Oriol Nieto, Jon Forsyth, Rachel M Bittner, and Juan P Bello. JAMS: a JSON annotated music specification for reproducible MIR research. In *International Society for Music Information Retrieval Conference*, ISMIR, 2014.