MEI2JSON: a pre-processing music scores converter

Charbel El Achkar* and Talar Atéchian

TICKETLAB,
Antonine University,
Baabda, Lebanon
Email: charbel.elachkar@ua.edu.lb
Email: talar.atechian@ua.edu.lb
*Corresponding author

Abstract: Converting music score content from symbolic formats to simplified data formats is found useful for artificial intelligence purposes. The conversion can be applied using XSL stylesheets and ontologies to ensure the preserving of the data quality throughout the transformation. In this paper, we proposed a new converter capable of transforming music scores encoded in MEI to JSON format for pre-processing purposes, and future usage into artificial intelligence techniques. The proposed converter uses an eastern music score ontology capable of structuring standard music scores content in addition to elements and attributes specific to eastern music. Thus, the converter shares the same support for eastern music scores. We illustrate the conversion process by assessing the performance analysis, the data quality, and the storage of the proposed converter in comparison with a combined approach composed of two state-of-the-art converters.

Keywords: MEI; MEI2JSON; music scores converter; MusicPatternOWL; eastern music.

Reference to this paper should be made as follows: El Achkar, C. and Atéchian, T. (2022) 'MEI2JSON: a pre-processing music scores converter', *Int. J. Intelligent Information and Database Systems*, Vol. 15, No. 1, pp.57–77.

Biographical notes: Charbel El Achkar is currently a PhD student at the University of Franche-Comté in France. He started his studies in October 2020 through a collaboration between the Antonine University in Lebanon and the University of Franche-Comté in France. His research interests are related to music analysis, generation and evaluation through artificial intelligence techniques. He had published his first research paper in July 2020 at the WIMS 2020 conference. He currently works as a software engineer at inmind.ai located in Beirut, Lebanon and Abu Dhabi, UAE while pursuing his research studies.

Talar Atéchian is an Associate Professor at the Faculty of Engineering at the Antonine University. She received her PhD from the INSA of Lyon, France in 2010. Her doctoral research focused in optimising routing protocols in vehicular ad-hoc networks. Currently, her research interest includes audio data analysis through artificial intelligence techniques.

This paper is a revised and expanded version of a paper entitled 'Supporting music pattern retrieval and analysis: an ontology-based approach' presented at 10th International Conference on Web Intelligence, Mining and Semantics, Biarritz, France, 30 June–3 July 2020.

1 Introduction

Combining artificial intelligence (AI) techniques with software solutions was found interesting for researchers and developers in the recent decade. The usage of AI helped in providing digital assistance as well as handling repetitive jobs for employees in their daily tasks. It helped with digital platforms where the need to reduce errors is one of the most essential and challenging criteria to improve its performance and reliability. Studies went deeper until they reached music-related interests. Many researchers and musicians took benefit of AI in their music-related studies. The latter provided digital assistance in music annotation platforms such as predicting the next note of a real-time annotated music score or generating new music scores depending on a pre-defined dataset.

However, both, the prediction of the next note and the generation of an entire music score require a well-defined pre-processing process to prevent data loss and reach higher accuracy post-training. This process and especially in music-related fields consist of applying several progressive tasks, such as finding the needed elements and attributes of a music score, filter the music score upon the use case, and finally, reshape the data and convert it to a specific format for training ingestion.

A platform for encoding and analysing eastern music scores named traditional modal monodies encoder (MM analyser) in Asmar et al. (2018) was capable of encoding a corpus specific to modal monodies of the Mashreq. The MusicPatternOWL ontology proposed in El Achkar and Atéchian (2020) assisted in the analysis process of the encoder ensuring errorless export of eastern music scores encoded in MEI format. The results of both, the encoder and the ontology, encouraged the use of resultant music scores in machine learning use cases, by the fact that they provide ready-to-ingest eastern music scores in MEI format.

While gathering the music scores out of the MM analyser, it was found that the MEI format is not the optimal format used to feed AI models. MEI is an XML-based format that holds multiple elements, each element gathers multiple music-related attributes to encode detailed music score content. This is where we highlight the need to convert the MEI outputs to simplified formats to reach our target of applying AI techniques on music score content.

Based on a related work investigation, we found that the MEI format can be converted to multiple formats such as MIDI and MusicXML. The latter formats were similar to the MEI in the matter of providing simplified data to the AI models. MusicXML is also an XML-based format and MIDI represents only recorded and played audio information. Further investigations led us to discover the MusicJSON format proposed in Alvaro and Barros (2010) capable of converting MusicXML music scores to JSON. JSON is an easy-to-use data ingestion format over XML. Its improved readability and lightweight approach support a bigger amount of information for feeding the AI models.

The MEI to MusicXML converters use the MEI encoding tools (https://github.com/music-encoding/music-encoding) provided by the MEI community for applying MEI conversions. These tools lack encoding and representing eastern music scores elements and attributes. Thus, the usage of existing MEI to MusicXML converters at the first stage, and the conversion of the resultant MusicXML outputs to JSON format at a second stage, generate JSON data that does not support eastern music score content.

In this paper, we present a new data converter named MEI2JSON that aims to convert the music scores encoded in MEI to JSON format while preserving their eastern music score content. The converter is based on the MusicPatternOWL ontology proposed in El Achkar and Atéchian (2020), in addition to a modified schema of MEI proposed in Asmar et al. (2018) capable of providing a structured knowledge extraction of music scores elements and attributes for eastern music encoded in MEI. The MEI2JSON is also capable of providing an MEI to JSON conversion without the need to combine multiple converters from multiple sources.

The remainder of this paper is organised as follows: In Section 2, we discuss the recent music related ontologies, XML to OWL mapping frameworks, and converters. In Section 3, we introduce the MEI2JSON converter, describing its main components, their behaviour, and the role of the MusicPatternOWL ontology inside these components. Section 4 explores the full implementation of the proposed converter through its application on an eastern music scores dataset encoded in MEI. In Section 5, we compare through experiments the proposed converter with a combination of two existent converters, followed by a conclusion and future work thoughts in Section 6.

2 Related work

Numerous studies were proposed to develop and manage ontologies related to music score contents. Jones et al. (2017) developed an ontology to semantically annotate and to reason upon western music scores. The proposed ontology helped in exploring the benefits of the web ontology language (OWL) in music-related fields. Also, due to the need of extracting the knowledge out of music data and managing this extraction process, an ontology took place in Cherfi et al. (2017) to integrate semantic music elements. This work helped in normalising the representation of music theories in a way they can be linked together.

Studies went extensive in the music field, especially when both developers and musicians found fruitful results in their collaborative opportunities. We proposed an ontology named *MusicPatternOWL* (El Achkar and Atéchian, 2020) to cover the structural and behavioural aspects of a pattern analysis algorithm for encoding eastern music scores. This ontology supports the semantic information retrieval and analysis processes of music score contents. The paper presented a proof of concept of its usage with an algorithm proposed in Abou Mrad (2016) and developed in Asmar et al. (2018) for analysing and encoding traditional modal monodies of the Mashreq, a unique corpus in eastern music.

Many music scores are usually encoded using symbolic formats such as MEI (Roland, 2002) and MusicXML (Good, 2001). These formats and especially MEI, are XML-based formats relying on XML schemas to describe the structure of their elements and attributes. This is where frameworks like JXML2OWL took place in Rodrigues et al. (2008) to manually map XML schemas to existing OWL ontologies

and later, automate the transformation of XML instances into individuals of the mapped ontology. These frameworks helped efficiently transforming the syntactic representation of data (using XML) to a semantic one (using OWL). This transformation provided the ability to perform inference on a knowledge-based model for better data exchange and integrity. Another mapping solution was to develop (Lacoste et al., 2011) an efficient framework of generating ontologies automatically out of XML instances. This framework helped in creating a good description of the OWL model and XML instance files. The introduction of both manual and automatic mapping frameworks (between OWL ontology and the XML schema) allowed accessing XML encoded data from Semantic Web applications that are already connected to OWL ontologies. This is where frameworks like SPARQL2XQuery took place to accentuate the adjacency and interoperability of both OWL and XML. Therefore, the proposed framework (Bikakis et al., 2009) was able to evaluate SPARQL queries over XML data after mappings XML to OWL Schemas.

Mapping frameworks were inspirational especially when the transformation rules between XML and OWL Schemas could be saved and reused upon demand by storing them in XSL stylesheets. The usage of XSL files as the holder of mapping rules encouraged their usage in multiple data formats converters, by the fact that they will ensure data conversion without losing data quality through the direct mapping between schemas in terms of datatypes and property rules. As for music-related researches, a toolkit named *music21* took place in Cuthbert and Ariza (2010) to provide software tools for both musicians with little programming experience and to programmers for analysing, searching and transforming music scores in symbolic forms. This toolkit did not use XSL stylesheets but provided several conversions supports to its specific format. This project supports conversion of several symbolic formats including MEI, MusicXML, and MIDI (The MIDI Manufacturers Association, 1995). By the evolution of the MEI format, Verovio, a music engraving library was developed in Pugin et al. (2014) to provide a visual representation of music scores encoded in MEI into SVG. This library also provided the capability to convert MusicXML to MEI and vice-versa based on the MEI XSL stylesheets available on the MEI encoding tools on Github (https://github.com/music-encoding/music-encoding).

The conversion via Verovio had limited capabilities, it focused on the main elements and attributes of music scores while excluding others. Also, an MEI related conversion framework named *Meico* took place in Berndt et al. (2018) to provide a novel tool that process MEI encoded music scores. *Meico* helped in converting MEI data to multiple symbolic formats like MusicXML. The conversion was based on the same XSL stylesheets used in Verovio where it also lacked the conversion of all the elements and attributes of a music score encoded in MEI. Another study presented in Alvaro and Barros (2010) focused on developing a music composing system named *Computer Music Cloud* (CMC) as well as a suitable data representation format named *MusicJSON* to efficiently compose and store music scores in the computer music cloud. The *MusicJSON* was considered as a music interchange tool between different services of the CMC. It was used also as a music data unification tool by converting several symbolic formats including MusicXML to a music representation format in JSON.

3 The MEI2JSON converter

3.1 Motivation

The MM analyser in Asmar et al. (2018) and the *MusicPatternOWL* ontology in El Achkar and Atéchian (2020) treated one of the most primary problems in music-related platforms. The latter is the lack of support for eastern music encoding and analysis. The MM analyser helped in encoding and analysing eastern music scores, and the *MusicPatternOWL* assisted in that analysis process by ensuring an errorless knowledge extraction at each progressive step of the encoding. At this stage, we were able to export lossless music scores encoded in MEI format.

The advantages of combining AI techniques with music-related platforms (presented in Section 1) motivated us to integrate those techniques and improve the MM analyser. Similar to any AI use case, the data must be prepared and simplified as much as possible before its training ingestion in neural networks. Therefore, it was needed to convert our MEI exports to another data format by the fact that MEI holds many elements and attributes that can be reduced upon the use case. Based on the music-related converters presented in Sections 1 and 2, the absence of a converter capable of transforming MEI music scores into JSON format was noticed, in addition to one of the essential criteria in question: converting music scores without losing data quality and preventing errors.

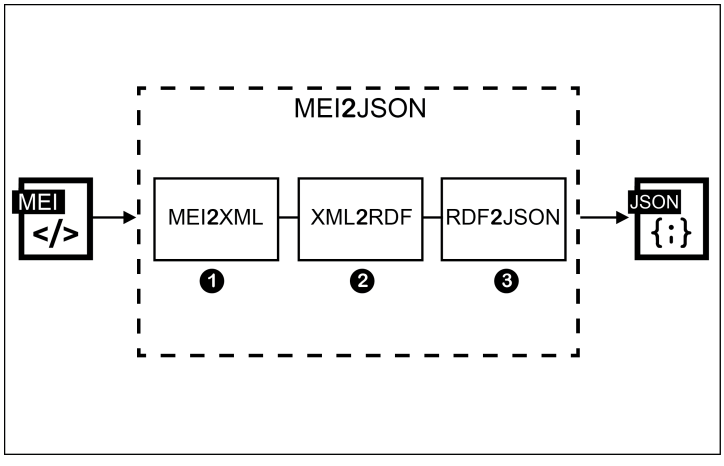
All the reasons mentioned above motivated us to create the MEI2JSON converter capable of transforming MEI music score to a simplified JSON format while preserving data quality and reducing data manipulation errors, especially for eastern music score datasets.

3.2 MEI2JSON components

By definition, “MEI is a community-driven, open-source effort to define a system for encoding musical documents in a machine-readable structure.” Its schema is developed using a literal programming XML format and expressed using the Relax NG (RNG) schema language. This music representation format and other primary ones focus on supporting occidental music because of its major worldwide usage. Therefore, it is not accurate for encoding eastern music scores as mentioned in Asmar et al. (2018) and El Achkar and Atéchian (2020). As stated in Sections 1 and 2, our need is to obtain the most simplified format out of eastern music scores encoded in MEI for future AI usage. The absence of a converter capable of handling eastern music elements and attribute at a first step, and the disability to convert MEI music scores to a simplified format using a single converter at a second, led us to create the MEI2JSON converter.

The MEI2JSON converter consists of three main components. Each component is responsible for a specific task to achieve successful MEI to JSON music scores conversion. As illustrated in Figure 1, any MEI in question should enter the MEI2XML component at a first phase, redirect the result of the first component to the XML2RDF component at a second phase, and at last convert the RDF data to JSON through the RDF2JSON component.

Figure 1 MEI2JSON main components overview



3.2.1 The MEI2XML component

As mentioned earlier, MEI is an XML-based format expressed using the RNG schema language. Thus, the MEI2XML component re-structures the schema of the MEI, producing a simplified XML output for the second component. Inside the MEI2XML, we use the modified MEI schema proposed in Asmar et al. (2018) to keep track of all the schema structure proposed by the MEI community, in addition to the elements and attributes proposed in Abou Mrad (2016) and contributed to the MEI schema in Asmar et al. (2018). These elements and attributes are essential when analysing and encoding eastern music scores. For this purpose, we configured an XSL stylesheet to embed our MEI to XML transformation rules. These rules hold structuring aspects to make the XML output the simplest possible. The proposed method consists of converting the RNG schema of MEI to a legacy XML schema. This consists of transforming the attributes of the MEI elements to sub-elements of the element itself and filters the concluding in the most optimal way possible.

After running the XSL stylesheet over several MEI files, we found that the modified MEI schema and the custom rules configured, lack consistency and normalisation. The configuration of custom rules resulted in different forms of output for a single MEI music score. Thus, we perceived the need to replace our custom rules with the *MusicPatternOWL* ontology proposed in El Achkar and Atéchian (2020). The *MusicPatternOWL* contains all the rules and restrictions needed to structure a music score encoded in MEI. It supports the same elements and attributes existent in the modified MEI schema, in addition to its power to extract and preserve the semantic information in an errorless manner.

Based on the XML to OWL frameworks mentioned in Section 2, the mapping between XML schemas and OWL schemas can be build using two different approaches. In case the OWL schema is existent, the XML and OWL schemas should be mapped manually, and in case the OWL schema does not exist, the OWL schema can be generated out of the existent XML schema, and by that, obtain an automatic mapping between them. Both, manual and automatic mapping approaches are used to transform the XML instances into OWL individuals. On the other side, our approach was not to

transform MEI to OWL individuals directly but to transform them to XML instances with OWL rules included, to structure and filter the needed data and exclude irrelevant ones. Since the *MusicPatternOWL* is inspired by the MEI schema and shares the same contribution as the modified MEI schema proposed in Asmar et al. (2018), a half-way mapping was already established. As for the MEI2XML component, we completed this mapping process by configuring an XSL stylesheet holding all the necessary mapping and transformation rules to convert an MEI music score into a simplified XML format.

The mapping rules are classified into three distinct types:

- class mapping
- datatype property mapping
- object property mapping.

The class mapping concerns creating a link between a node of the modified MEI schema with an OWL concept of the *MusicPatternOWL* ontology. The datatype property mapping links an MEI node to a datatype property of the *MusicPatternOWL*. The object property mapping relates two-class mappings to an OWL object property of the *MusicPatternOWL*.

As for the transformation rules, in addition to the ones embedded through mapping, we note the re-structuring shown in the XML representation below, to obtain the optimal XML output possible. The first representation is a measure of an MEI score entered as input, and the second one is the same measure converted to the XML output through the configured XSL stylesheet. Since the *MusicPatternOWL* ontology excludes meta-data features, the latter is excluded from the XML output generated, due to the absence of mapping between the MEI schema and the *MusicPatternOWL* for this purpose. This feature is found essential by the fact that it can automatically filter irrelevant data, focusing on the music-score itself to achieve a successful pre-processing process.

At this stage, the MEI2XML component relies on an XSL stylesheet capable of transforming MEI scores to XML format, while preserving music elements and attributes specific to eastern music.

```
               The measure representation in an MEI file
<measure xml:id="m-32" label="1" left="rptstart" n="1">
  <staff xml:id="m-34" n="1">
    <layer xml:id="m-35" n="1">
      <beam xml:id="m-37">
        <note xml:id="m-36" dur="8" dur.ges="128p" oct="4" pname="d"
        pnum="50" stem.dir="up" snr="\alpha"/>
        <note xml:id="m-38" dur="8" dur.ges="128p" oct="4" pname="g"
        pnum="55" stem.dir="up" snr="\beta" mnr="yes"/>
        <note xml:id="m-40" dur="8" dur.ges="128p" oct="4" pname="f"
        pnum="54" stem.dir="up">
          <accid xml:id="m-41" accid="s"/>
        </note>
      </beam>
    </layer>
  </staff>
  <tie xml:id="m-39" endid="#m-40" startid="#m-38"/>
</measure>
```

The measure representation in the XML output

```
<measure number="1">
  <beam>
   <note>
      <pname>d</pname>
      <oct>4</oct>
      <snr>\alpha</snr>
      <dur>8</dur>
   </note>
   <note>
      <pname>g</pname>
      <oct>4</oct>
      <snr>\beta</snr>
      <mnr>\beta</mnr>
      <dur>8</dur>
   </note>
   <note>
      <pname>f</pname>
      <oct>4</oct>
      <dur>8</dur>
      <accid>s</accid>
   </note>
  </beam>
</measure>
```

3.2.2 The XML2RDF component

The usage of XSL stylesheet in the MEI to XML conversion of the first component encouraged us to take the same approach in the next one. The objective of the XML2RDF component is to convert the XML data into RDF without losing any semantic information. Therefore, we decided to use the XSL stylesheet proposed in Breitling (2009). The latter contains all the standard transformation rules capable of providing efficient XML to RDF conversion. In other terms, we can apply this converter to any XML dialect which supports then both, the elements and attributes proposed in the modified MEI schema in Asmar et al. (2018). At this stage, we were able to convert MEI scores to XML using MEI2XML and convert the XML to RDF using the XML2RDF component, without losing any semantic information related to eastern music scores.

Note that mentioning the support of eastern music scores does not eliminate the fact that music encoding formats were initially built to support occidental music scores. Therefore, the MEI2JSON converter supports the latter if encoded in MEI format.

3.2.3 The RDF2JSON component

The previous components of the MEI2JSON converter managed to convert MEI scores to RDF using several methods to prevent loss of semantic information and data quality. Thus, the job of the RDF2JSON component is to proceed with the conversion process to convert the music score encoded in MEI to JSON format. As mentioned in Section 1, the MEI2JSON converter aims to transform MEI files to JSON for pre-processing purposes. The pre-processing process, in addition to data cleaning and filtering, consists

of applying feature engineering selection to choose the needed input variables for training ingestion. In music-related cases, these input variables are the elements and attributes of a music score, where we must select the needed ones only, to solve targeted use cases. As an example, when the use case is to predict the next note of a music score, we must select the input variables (elements and attributes) that affect only the note element of a music score. Therefore, we use the *MusicPatternOWL* ontology proposed in El Achkar and Atéchian (2020) as element selector and validator in the RDF2JSON component. This way we can produce the most optimal JSON output by selecting the needed elements from the music score upon the use case, validating once again the three mapping types of the MEI2XML component, and finally building the JSON output to achieve a successful pre-processing process.

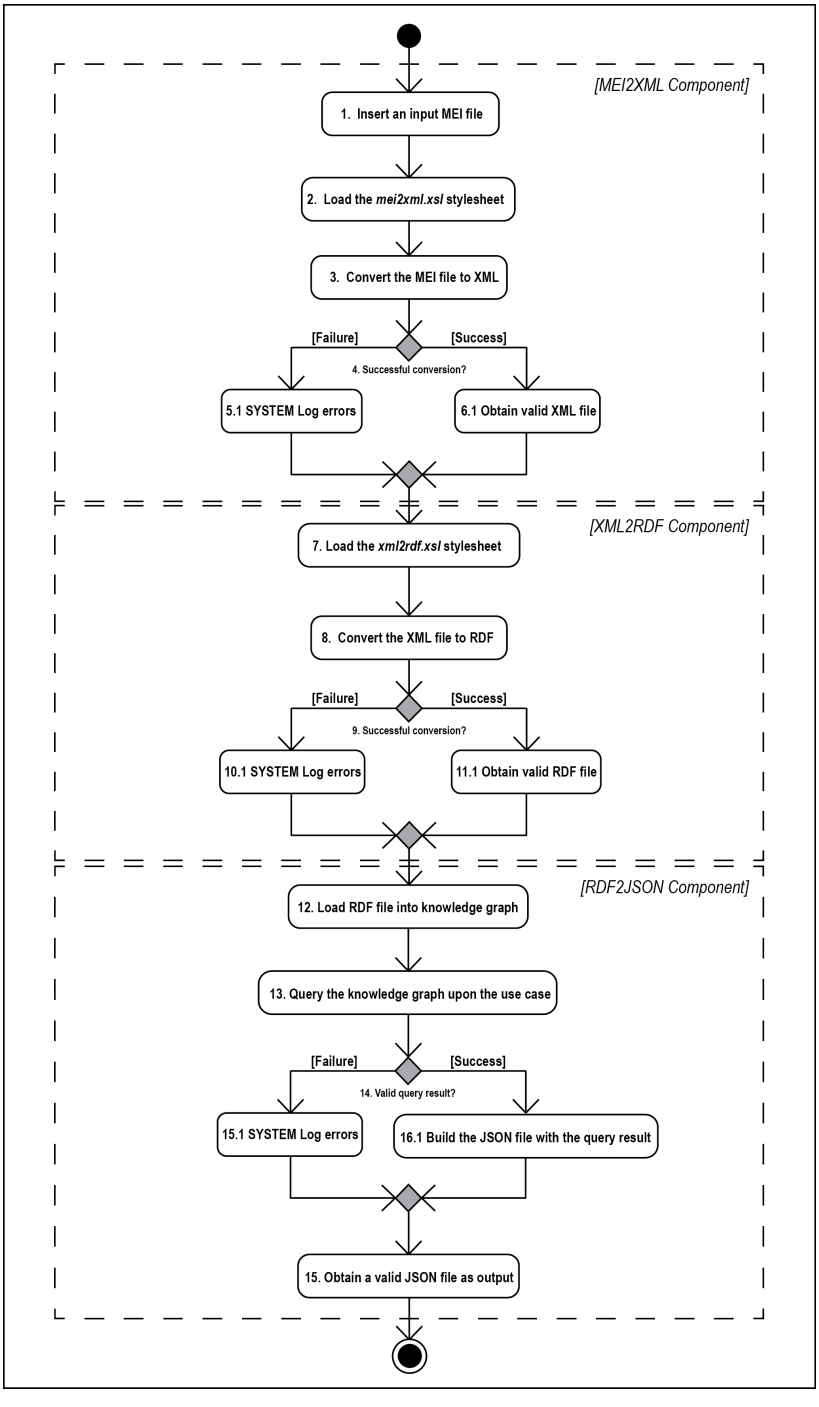
It is important to mention that this component excludes the attributes of a music score since the MEI2XML component transforms the attributes to sub-elements as shown in the earlier XML representation. Also, the RDF2JSON component contains a knowledge graph builder so that using SPARQL queries the *MusicPatternOWL* is capable of selecting the needed features through a simple query builder. Note that the query result is passed through JSON libraries to ensure the creation and validity of the output.

To recapitulate and converge, the MEI2JSON relies on three components. The first component, the MEI2XML, converts the structure of an MEI music score to XML by transforming its schema represented in RNG to the XML schema. Its conversion relies on a mapping between the modified MEI schema presented in Asmar et al. (2018) and the *MusicPatternOWL* proposed in El Achkar and Atéchian (2020). This mapping is implemented in an XSL stylesheet, the core of the MEI to XML conversion. The second component, the XML2RDF, uses the existing XSL stylesheet proposed in Breitling (2009) for converting XML to RDF. Since this stylesheet supports any XML dialect, we only configured this component to reach the RDF format for input in the last component. The third and last component, the RDF2JSON component uses the *MusicPatternOWL* as a music score validator ensuring a lossless flow of information. It converts RDF data to JSON while implementing the idea of query builder where users can filter and retrieve their needed music elements upon future AI use cases. Therefore, the unification of these components constitutes the MEI2JSON capable of transforming eastern music scores to a ready-to-ingest format in AI models.

4 Implementation

The previous section presented each component of the MEI2JSON converter. It exposed the role, the composition, and the benefit of each component to achieve a successful conversion of music scores encoded in MEI to JSON output. The present section exposes the necessary technical details to achieve the full implementation of the proposed converter, in addition to the implementation of two combined converters for further experimental comparison.

Figure 2 MEI2JSON activity diagram



4.1 MEI2JSON process

Figure 2 presents the MEI2JSON converter through an activity diagram. The first part of the diagram illustrates the progressive steps of the MEI2XML component to achieve successful conversion (MEI to XML). The converter pulls an MEI score taken from an MEI file (*.mei* extension), loads the custom XSL stylesheet created and converts if possible, the MEI file to XML. The custom XSL stylesheet named *mei2xml.xsl* is the file responsible for handling the conversion needed. Thus, the *mei2xml.xsl* needs to be loaded by an XSLT processor to perform this conversion. Therefore, we use the Saxon XSLT and XQuery processor (Kay, 2010) based on its previous usage in most of the converters presented in Section 2. In the case of a successful conversion, the generated XML file will be redirected to the second component to perform further steps. Otherwise, the system logs the errors so that we can easily find and solve the problems related to the failure in conversion.

The generated XML file proceeds its path to the XML2RDF component. Like the previous component, the *xml2rdf.xsl* proposed in Breitling (2009) is loaded using the Saxon processor to apply the corresponding conversion to the XML file. Also, the generated RDF file proceeds to the next component in success cases, and in case of failure, the error loggings will guide the user to solve the problems faced.

Finally, the generated RDF file is loaded in the RDF2JSON component using the library (RDFLib, https://github.com/RDFLib/rdflib). This library provides powerful parsers and serialisers to load the knowledge graph out of RDF/XML data. Once loaded, the RDF file can be queried through custom SPARQL queries to extract the semantic information needed for pre-processing purposes. The SPARQL query then can be customised upon the use case. In case of a successful query, the result will be sorted and formed in a JSON file as output. The RDF2JSON component ensures the validity of the JSON file by applying schema syntax definitions such as the JSON schema proposed in Pezoa et al. (2016). Thus, the MEI2JSON made several progressive steps passing from a component to another, to achieve a successful conversion of MEI scores to JSON.

Note that the MEI2JSON converter is currently implemented using the Python language (Van Rossum and Drake, 2009), although, it can be implemented using other programming languages since we are loading the XSL stylesheet through command-line usage of the Saxon library. Also, the RDF related libraries are available in many programming languages which helps in providing enhanced coverage of the MEI2JSON converter.

4.2 Meico+MusicJSON process

The related work presented all the converters capable of transforming MEI scores to other symbolic formats such as MusicXML or MIDI. Also, it was mentioned that the music community does not have a straightforward approach to convert MEI scores to JSON.

In this part, we present the usage of two different converters so that once implemented, they can be used in combination to assess the MEI2JSON converter. The first converter named *Meico* can convert MEI scores to MusicXML, and the second one, the *MusicJSON* converter can convert MusicXML scores to JSON format.

Figure 3 Meico+MusicJSON activity diagram

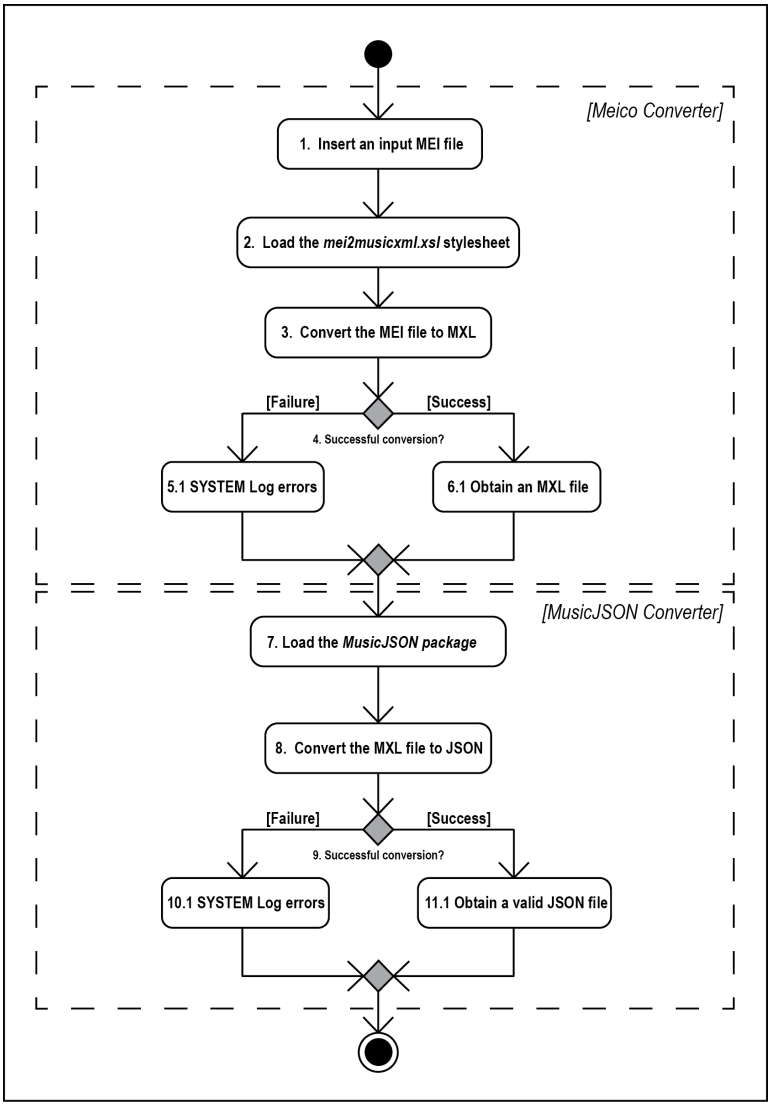


Figure 3 presents the combined converters through an activity diagram. The first portion of this diagram concerns the insertion of the MEI score as input to the *Meico* converter, loading the *mei2musicxml* stylesheet provided by MEI encoding tools (https://github.com/music-encoding/music-encoding), and converting the MEI score to MusicXML format. In case of a successful conversion, the obtained MusicXML file proceeds to the second converter. Otherwise, the system logs the errors found while converting to detect and solve related problems. The second portion of this diagram concerns the insertion of the MusicXML file generated by the *Meico* converter and converting this file to JSON format using *MusicJSON* proposed in Alvaro and Barros (2010). In successful cases, the result will be a valid JSON output that respects the schema proposed by the *MusicJSON* contributors.

It is important to mention that both, the *Meico* (Berndt et al., 2018) and *Verovio* (Pugin et al., 2014) converters rely on the same XSL stylesheet provided by MEI encoding tools (https://github.com/music-encoding/music-encoding) to run the transformation over MEI scores and convert them to MusicXML. Also, they use the same approach of using the command-line interface to apply this conversion which makes them identical in this matter. Therefore, using *Meico* with *MusicJSON* as the combined approach or using *Verovio* with *MusicJSON* will result in the same experimental results in terms of data quality and complexity metrics.

Note that the *MusicJSON* converter consists of a package written in JavaScript programming language, loaded using the Node.js runtime environment to achieve the corresponding conversion. This package, in addition to the whole process, is called using Python programming language (Van Rossum and Drake, 2009) for better comparison with the proposed converter.

5 Experiments

In the implementation section, we presented both the MEI2JSON converter and the two combined converters *Meico+MusicJSON*. We elaborated the two processes using activity diagrams to technically describe the role of each component inside both approaches. In this section, we aim to compare the MEI2JSON with the *Meico+MusicJSON* in terms of performance analysis and the quality of the data produced out of these converters. For this purpose, we use a dataset of 150 traditional modal monodies music scores encoded in MEI. These music scores are considered a unique corpus in eastern music. Also, these MEI scores are the output of the MM analyser, the platform proposed in Asmar et al. (2018) to analyse and encode eastern music. Thus, the music scores contain elements and attributes specific to eastern music analysis, in addition to the standard elements and attributes present in any music score encoded in MEI.

5.1 Dataset

Considering that our primary objective is to provide successful MEI to JSON conversion of eastern music scores, we chose a dataset related to traditional modal monodies. Modal monodies are eastern music scores, thus the dataset used in our experiments contains the following four eastern music modes:

- Hijāz (31 music scores)
- Dūlab Bāyāti (33 music scores)
- Dūlab Rāst (40 music scores)
- Jāhārkā (46 music scores).

Once grouped, we obtain 150 eastern music scores encoded in MEI format. The size of an MEI score varies between 4.4 to 42.6 kB of music score data. Musicians transcribe modal monodies from eastern music score books such as Abou Mrad (2016) to MEI format. They encode and validate their digitalised transcriptions using the MM analyser, and provide us with the needed MEI scores for further studies.

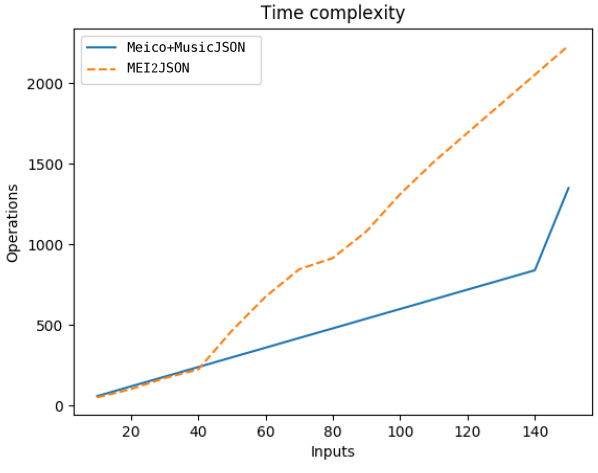
5.2 Performance analysis

The performance analysis concerns analysing algorithms based on an input size required to run it. The complexity then is expressed as a function of n, where n is the input size. In this paper, we compare the MEI2JSON with the *Meico+MusicJSON* in terms of two complexity metrics, the time and space complexity. The time complexity describes the amount of time to run an algorithm, and space complexity reports the amount of memory space to run an algorithm. We calculated the time complexity through experimental evaluation, and used the *memory-profiler* python module to evaluate the space complexity for both approaches.

5.2.1 Time complexity

Figure 4 presents the time complexity chart for the MEI2JSON and the *Meico+MusicJSON* converters. We use an orange-dashed line to visualise the MEI2JSON converter, and a blue line to illustrate the *Meico+MusicJSON* converter. The two approaches use the same number of input (150 music scores) and the same number of elementary operations performed by the algorithm for better comparison purposes. Figure 4 shows the same performance of both approaches when the input size is smaller than 40 music scores. However, the time complexity changes clearly after reaching a value of 60 music scores, taking a different trajectory for each approach.

Figure 4 Time complexity chart (see online version for colours)



The algorithm of the *Meico+MusicJSON* is not distinctly affected by the size of each music score. On the other hand, the proposed MEI2JSON converter lacks this stability due to his final component, the RDF2JSON component. The usage of the RDF format for data selection and second validation through the *MusicPatternOWL* forces the algorithm to load the entire RDF file inside the knowledge graph of the RDF2JSON component. The latter slows the converter dependent on the size of the RDF file in question, the time to load the RDF inside the knowledge graph, and the selection of the needed elements using SPARQL queries.

Therefore, the *Meico+MusicJSON* outperforms the MEI2JSON in terms of time complexity due to the dependency of the latter on the size of each music score. Note that the number of operations of the *Meico+MusicJSON* augmented quickly after exceeding a value of 140 music scores as input size. This unpredicted augmentation could result in a draw between the two approaches when using bigger datasets.

5.2.2 Space complexity

Regarding the space complexity, we use the *memory-profiler* python module to monitor the memory consumption of both the *Meico+MusicJSON* and the MEI2JSON converters. Figures 5 and 6 present the memory consumption (in MiB) expressed as a function of time to respectively estimate the space complexity of *Meico+MusicJSON* and MEI2JSON. The space complexity in both graphs is calculated by running the *memory-profiler* on the algorithms using the entire dataset of 150 music scores.

In Figure 5, we can visualise an approximate constant line with a value of 48.0 MiB during the whole process. We interpret the chart by the fact that the *Meico* converter start by loading the *mei2musicxml.xsl* stylesheet to convert the MEI score to MusicXML. This loading allocates an amount of memory equal to 48.0 MiB until the conversion completes. Once completed, the *MusicJSON* converter allocates an amount of 48.7 MiB to load its package and convert the MusicXML scores to JSON format (from 310 to 330 millisecond in Figure 5). The entire process took 330 milliseconds to convert the 150 music scores to JSON.

As for Figure 6, we can visualise two main variations of memory consumption. The first is a continuous line taking a value of 48.0 MiB from the beginning till a time equal to 380 milliseconds. The second is an approximate line to 72.0 MiB from 380 milliseconds to the end (630 milliseconds). The first line stable on 48.0 MiB concerns loading the custom *mei2xml.xsl* stylesheet responsible of converting the MEI scores to XML format. Once the first conversion completes, the same memory consumption is given to load the *xml2rdf.xsl* stylesheet responsible for converting the XML format to RDF. Both stylesheets use the processor proposed in Kay (2010), which explains the fact that they have the same memory consumption (from 0 to 380 milliseconds). Therefore, the MEI2XML and XML2RDF components allocate the same amount of memory.

Once the role of the XML2RDF component completes, the line chart varies to reach a value stable on 72.0 MiB approximatively, to highlight the third component, the RDF2JSON. The load of the RDF results using the proposed library (RDFLib, https://github.com/RDFLib/rdflib) is responsible for reaching this memory consumption value. This library handles the loading of RDFs into the knowledge graph at first, and querying the needed elements out of the RDF file loaded in second. The entire process took 630 milliseconds to convert the 150 music scores to JSON.

Note that MEI2JSON's last component is the part responsible for increasing the time and memory consumption in comparison with the *Meico+MusicJSON* approach. Therefore, the RDF2JSON component is responsible for increasing the space complexity and the time complexity as seen in the previous interpretations. Improving this component in the future can make the proposed converter outperform the two combined ones in terms of time and space complexity.

Figure 5 Space complexity chart – Meico+MusicJSON (see online version for colours)

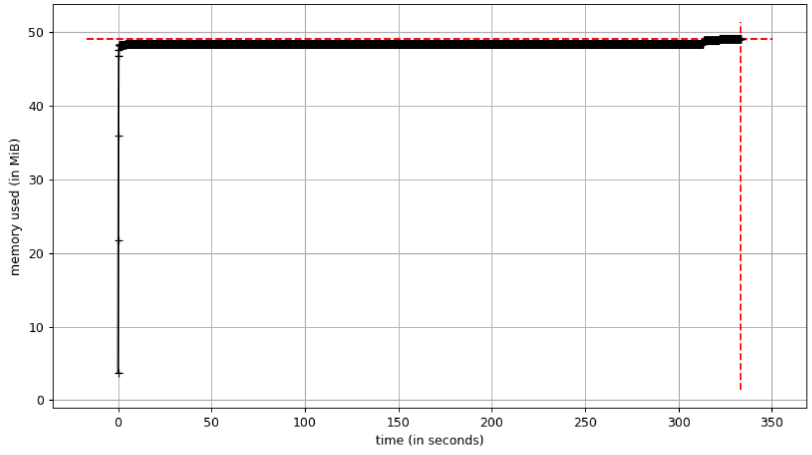
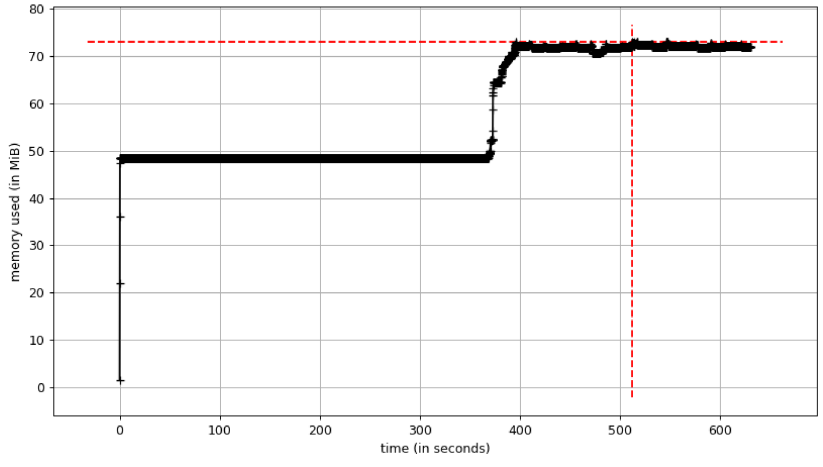


Figure 6 Space complexity chart – MEI2JSON (see online version for colours)



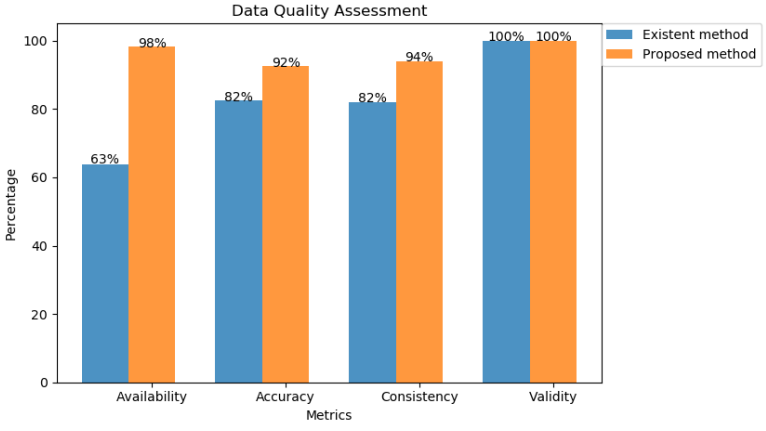
5.3 Data quality assessment

The performance of converters is usually evaluated by calculating its complexity and ensuring it preserves the quality of the data produced out of its transformation. In Figure 7, we present four different data quality metrics used on both the *Meico+MusicJSON* and the MEI2JSON converters to assess their quality preserving upon the used dataset. The MEI2JSON metrics are visualised using orange bars in the histogram, and the *Meico+MusicJSON*'s using blue bars.

Before explaining the quality metrics, we note the usage of the jsonix (https://github.com/highsource/jsonix) mapping library to obtain a JSON schema out of the modified MEI schema proposed in Asmar et al. (2018). Thus, we used the resulted JSON schema and the *MusicJSON* schema to respectively evaluate the output of the *Meico+MusicJSON* and the MEI2JSON converters upon each metric. It is valuable to

mention that the mandatory elements in this experiment concern the note element and its attributes, including the eastern music score ones.

Figure 7 Histogram – data quality metrics (see online version for colours)



Below we present the assessment metrics to compare both approaches:

- *Availability* is a metric used to measure whether all the necessary elements of a music score are present in a specific dataset. The dataset in this matter concerns the output generated out of both converters. We measure the availability of both approaches by calculating the percentage of music score fields that have values entered into them. The MEI2JSON resulted in an availability percentage of 98.2% and the *Meico+MusicJSON* of 63.9%. The gap between both results is mainly due to the lack of support of the modified MEI schema in the *Meico+MusicJSON* approach. This lack is responsible for discarding undefined elements and preserving only the standard occidental music score elements. In this case, the undefined elements are the ones related to eastern music scores.
- *Accuracy* is a metric used to evaluate the correctness of the music score in question. We measure the accuracy of both approaches by calculating the percentage of the correctly converted music score elements compared to the initial values. Since the future usage of the converted music scores is in the AI field, we chose to estimate accuracy using the *accuracy_score* function provided by Pedregosa et al. (2011). Considering the usage of several essential elements for encoding music scores, we estimate the accuracy as a multilabel approach. We calculate the accuracy at the level of multiple elements, each element containing multiple labels. As shown in the equation below, the accuracy is the sum of accuracies at the level of each music element divided by N - the number of music score elements used. The *true_label* are the labels of the initial music elements before conversion and the *output_label* are the labels of elements after JSON or *MusicJSON* conversion. It is important to mention that since music score elements have labels encoded as characters in MEI, we had to use the *LabelEncoder* function provided by Pedregosa et al. (2011) to convert characters to numeric. This way the element's labels would be compatible for usage in the *accuracy_score* function.

For example, and before using the *accuracy_score* function, we use the *LabelEncoder* to transform the labels of the PitchName element from [a, b, c, d, e, f, g] to [0, 1, 2, 3, 4, 5, 6] so that we can calculate the accuracy of the PitchName using *accuracy_score* at first, calculate the accuracy of the remaining elements, sum all the accuracies and divide them by the number of elements used. Finally, we multiply the resulting accuracy by 100 to obtain the accuracy value as a percentage.

$$Accuracy = \frac{\sum accuracy\_score(true\_label, output\_label)}{N} \times 100$$

The *Meico+MusicJSON* resulted in an accuracy of 82.4%, and the MEI2JSON an accuracy of 92.5%.

- *Consistency* is a metric used to evaluate the synchronicity of music score in terms of data types and schema structure. We measure this metric by calculating the percentage of data types that match across different records. We use the schema structure of both approaches to detect the structure and data types changes while passing from a component/converter to another.

  Similar to accuracy calculation, we took the same approach to calculate the consistency at the level of data types per music score element. Therefore, we use the *LabelEncoder* function to transform data type values to numeric labels and use the *v_measure_score* function provided in Pedregosa et al. (2011) to estimate consistency of music score elements.

  By definition the *v_measure_score* clusters the labels given a ground truth. In our case, it clusters the element's data type labels reflecting the consistency of the latter elements. The *v_measure_score* function takes as parameters the following: The *true_label* which stands for the element's data type labels before using the MEI to JSON or *MusicJSON* converters. The *output_label* whichs stands for the element's data type labels after MEI to JSON or *MusicJSON* conversion. The last parameter, β, is the ratio of weight attributed to homogeneity and completeness. We will leave this value to its default meaning that the resultant score should have the same weight regarding homogeneity and completeness. The *v_measure_score* results in a score between 0.0 and 1.0. The greater the result, the better is the consistency.

  Once the *v_measure_score* is calculated against all the essential elements of a music score, we sum all the resultant scores, divide them by N - the number of music score element used. Finally, we multiply the whole result by 100 to obtain the final consistency value as a percentage.

$$Consistency = \frac{\sum v\_measure\_score(true\_label, output\_label, \beta)}{N} \times 100$$

  As shown in Figure 7, the consistency percentage of *Meico+MusicJSON* is equal to 82% and the MEI2JSON equal to 94%. This slight improvement of the MEI2JSON over the combined approach is due to the existence of the *MusicPatternOWL* ontology present in the first and last converter, to structure and filter the music score elements in question.

- *Validity* is a metric used to measure how well data conforms to the required value attributes. We measured the validity by calculating the percentage of music score elements and attributes that have values within the domain of acceptable values. We used the JSON schema of both approaches, in addition to the syntax definition proposed in Pezoa et al. (2016) to calculate the validity metric. Both approaches resulted in a validity percentage of 100%. This high percentage is due to *MusicJSON*'s built-in validator in the first approach and the presence of the *MusicPatternOWL* ontology in the second.

5.4 Storage assessment

The previous assessment reflected the outperforming of the MEI2JSON converter over the *Meico+MusicJSON* in terms of data quality metrics. The current assessment presents the storage reduction of both approaches at different input size scales.

Table 1 Storage overview table

Numbers of inputs (files)	*10*	*20*	*30*	*40*	*50*	*60*
Initial input size (in kB)	58.9	115.3	208.1	267.5	546.2	861.4
Meico+MusicJSON output size (in kB)	58.5	113.9	191.4	249.5	398.0	542.6
MEI2JSON output size (in kB)	55.9	100.7	146.9	181.4	224.7	259.1

70	*80*	*90*	*100*	*110*	*120*	*130*	*140*	*150*	*Reduction (%)*
1,100	1,200	1,500	1,900	2,200	2,500	2,800	3,100	3,300	——
658.7	704.3	829.9	1,000	1,200	1,300	1,400	1,500	1,700	45.3
305.2	325.4	346.5	414.3	450.2	497.0	542.6	595.6	645.8	74.1

In Table 1, we demonstrate the storage allocation of both conversion approaches using the same input size scale as the complexity study. The first row of the table presents the different scales of input sizes. The second corresponds to the initial size of the MEI music scores before conversion. The third and the last concern the output sizes of the *Meico+MusicJSON* and the MEI2JSON converter. The input and output sizes expressed in kiloBytes (kB). While assessing the storage, it was clear that the *Meico+MusicJSON* reduced the storage allocation depending on the input size. Therefore, we calculated the average reduction percentage that resulted in a decrease of 45.3%. On the other side, the MEI2JSON was able to reduce the storage with an average of 74.1%. Thus, the MEI2JSON is capable of reducing the storage by 28.8% more than the *Meico+MusicJSON* approach. This improvement is beneficial for database systems where it can ensure the integrity of music score using one of the most optimal format possible, the JSON format. Also, it has a positive influence on our pre-processing target, since ingesting smaller inputs makes the neural network handle bigger datasets while keeping the hardware in healthy conditions.

To briefly summarise our experiments, we calculated the time and space complexity, the data quality metrics, and the storage assessment of both converters using an eastern music dataset presented at the beginning of Section 5. The *Meico+MusicJSON* approach outperformed MEI2JSON in terms of time and space complexity. The latter outperformed *Meico+MusicJSON* in terms of data quality assessment and storage assessment. Thus, the MEI2JSON proved its role as a converter for eastern music scores

from MEI to JSON format where other converters focused primarily on supporting occidental music scores. On the other hand, the comparison between the previously mentioned converters will normally differ using occidental music scores. The existing solution should outperform the MEI2JSON in terms of data quality metrics since occidental music scores are in continuous development in the MEI community. These continuous updates are supported by the MEI converters like *Meico* (Berndt et al., 2018) since they use the latest versions of the MEI encoding tools (https://github.com/music-encoding/music-encoding).However, the MEI2JSON relies on the *MusicPatternOWL* proposed in El Achkar and Atéchian (2020) that supports eastern music scores more than the occidental because of the latter's continuous updates.

6 Conclusions

In this paper, we proposed the MEI2JSON converter that covers the transformation of music scores encoded in MEI to JSON format for pre-processing purposes. As explained, the proposed converter consists of three components. The components rely on the *MusicPatternOWL* ontology to achieve information retrieval and structure music score content throughout the conversion process. We compared the MEI2JSON with a combined approach composed of two existent converters, *Meico* and *MusicJSON*. We used a dataset of 150 eastern music scores encoded in MEI to obtain the needed results. The experiment results were promising by the fact that our converter was able to outperform the combined converters in terms of data quality and storage assessment. The converter proved its capability of preserving the quality of the data while reducing the allocated storage space. However, the combined approach still outperforms the MEI2JSON in terms of analysis performance. The outperformance was mainly due to the behaviour of the last component, the RDF2JSON component. Our future work should focus on improving the analysis performance of the MEI2JSON, in addition to deploying the proposed converter on several database systems to ensure its integrity upon diverse datasets.

References

Abou Mrad, A. (2016) *Éléments de sémiotique modale. Essai d'une grammaire musicale pour les traditions monodiques*, Éditions Geuthner et Éditions de l'Université Antonine.

Lvaro, J.L. and Barros, B. (2010) 'MusicJSON: a representation for the computer music cloud', *Proceedings of the 7th Sound and Music Computer Conference*, Barcelona.

Asmar, M., Atéchian, T., Martin, S.L. and Mrad, N.A. (2018) 'Traditional modal monodies generative grammar encoding in the music encoding initiative', *Proceedings of the International Conference on Technologies for Music Notation and Representation*, Concordia University, pp.95–103, DOI: 10.5281/zenodo.1289693.

Berndt, A., Waloschek, S. and Hadjakos, A. (2018) 'Meico: a converter framework for bridging the gap between digital music editions and its applications', *Proceedings of the Audio Mostly 2018 on Sound in Immersion and Emotion (AM'18)*, DOI: 10.1145/3243274.3243282.

Bikakis, N., Gioldasis, N., Tsinaraki, C. and Christodoulakis, S. (2009) 'Querying XML data with SPARQL', *DEXA 2009*.

Breitling, F. (2009) 'A standard transformation from XML to RDF via XSLT', *ArXiv abs/0906.2291*.

Cherfi, S.S-S., Guillotel, C., Hamdi, F., Rigaux, P. and Travers, N. (2017) 'Ontology-based annotation of music scores', *K-CAP 2017*.

Cuthbert, M. and Ariza, C. (2010) 'Music21: a toolkit for computer-aided musicology and symbolic music data', *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010*, pp.637–642.

El Achkar, C. and Atéchian, T. (2020) 'Supporting music pattern retrieval and analysis: an ontology-based approach', *Proceedings of the 10th International Conference on Web Intelligence, Mining and Semantics (WIMS 2020)*, Biarritz, France, pp.17–20, Association for Computing Machinery [online] https://doi.org/10.1145/3405962.3405973.

Good, M. (2001) 'MusicXML for notation and analysis', in Hewlett, W.B. and Selfridge-Field, E. (Eds.): *The Virtual Score: Representation, Retrieval, Restoration*, pp.113–124, MIT Press, Cambridge (MA); London (UK).

Jones, J., Braga, D., Tertuliano, K. and Kauppinen, T. (2017) 'MusicOWL: the music score ontology', *Proceedings of the International Conference on Web Intelligence*.

Jsonix [online] https://github.com/highsource/jsonix (accessed 23 October 2020).

Kay, M (2010) *Saxon the XSLT and XQuery Processor* [online] https://www.researchgate.net/publication/247443507_Saxon_the_xslt_and_xquery_processor (accessed 23 October 2020).

Lacoste, D., Sawant, K. and Roy, S. (2011) 'An efficient XML to OWL converter', *Proceedings of the 4th India Software Engineering Conference 2011, ISEC'11*, pp.145–154, DOI: 10.1145/1953355.1953376.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011) 'Scikit-learn: machine learning in Python', *Journal of Machine Learning Research*, Vol. 12, No. 12, pp.2825–2830.

Pezoa, F., Reutter, J.L., Suárez, F., Ugarte, M. and Vrgoc, D. (2016) 'Foundations of JSON schema', *WWW 2016*.

Pugin, L., Zitellini, R. and Roland, P. (2014) 'Verovio: a library for engraving MEI music notation into SVG', *Proceedings of the 15th International Society for Music Information Retrieval Conference*, Taipei, Taiwan, pp.107–112, ISMIR, DOI: 10.5281/zenodo.1417589.

RDFLib [online] https://github.com/RDFLib/rdflib (accessed 22 October 2020).

Rodrigues, T., Rosa, P. and Cardoso, J. (2008) 'Moving from syntactic to semantic organizations using JXML2OWL', *Computers in Industry*, Vol. 59, pp.808–819, DOI: 10.1016/j.compind.2008.06.002.

Roland, P. (2002) 'The Music Encoding Initiative (MEI 01 2002)', *Proceedings of the First International Conference on Musical Applications Using XML*, pp.55–59.

The MIDI Manufacturers Association (1995) *MIDI 1.0 Detailed Specification*, Document version 4.2, revised 1995, Los Angeles, CA.

The Music Encoding Initiative [online] https://github.com/music-encoding/music-encoding (accessed 22 October 2020).

Van Rossum, G. and Drake, F.L. (2009) *Python 3 Reference Manual*, CreateSpace, Scotts Valley, CA.