MEL: Metadata Extractor & Loader

Sergio J. Rodríguez Méndez[1,2][0000−0001−7203−8399], Pouya G. Omran[1,3][0000−0002−4473−3877], Armin Haller[1,4][0000−0003−3425−0780], and Kerry Taylor[1,4][0000−0003−2447−1088]

[1] Australian National University, Canberra ACT 2601, AU
[2] Sergio.RodriguezMendez@anu.edu.au
[3] P.G.Omran@anu.edu.au
[4] {firstname.lastname}@anu.edu.au
https://cs.cecs.anu.edu.au/

Abstract. The metadata and content-based information extraction tasks from heterogeneous file sets are pre-processing steps of many Knowledge Graph Construction Pipelines (KGCP). These tasks often take longer than necessary due to the lack of proper tools that integrate several complementary extraction methods and properties to get a rich output set. This paper presents *MEL*, a Python-based tool that implements a set of methods to extract metadata and content-based information from unstructured information encoded in different source document formats. The results are generated as JSON files, which can: (a) optionally be stored in a document store, and (b) easily be mapped to RDF using a variety of tools such as *J2RM*. *MEL* supports more than 20 different file types, making it a versatile tool that aids pre-processing tasks as part of a KGCP based on comprehensive configurable settings.

Keywords: Metadata Extraction · Information Extraction · Data Preprocessing · Knowledge Graph Construction · Data Analysis Pipeline

1 Introduction

This paper introduces *MEL*[5], a tool that implements a set of methods to extract metadata and content-based information from various file formats as JSON[6] objects. For each supported file type, *MEL* extracts the textual content from the source document and performs specific pre-processing and data cleaning tasks. Also, it performs basic text analysis tasks (pattern matching and keyword extraction) and generates the results in a machine-readable format (JSON), preparing

[5] The submission type of this paper is a demonstration that portraits the tool's general functionality. The demo presents: input document set, initial settings, processing, and output set. The input document set is artificial in order to show various capabilities of the tool for different file types and formats. All resources and demo videos are available at https://w3id.org/kgcp/MEL-TNNT

[6] https://www.json.org/

Table 1. *MEL*'s general structure of the JSON outputs

JSON Object	Description
`_id`, `_rev`	Internal identifier and revision number of the stored result file in the document store.
`General-Metadata`	Relevant file system attributes, such as file name, file size, file extension, and last modified date/time.
`Specific-Metadata`	File format's structure and properties of its "Document Object Model". For example, for `.msg` files (Outlook e-mails), *MEL* extracts the e-mail header data fields, body, and attachments (recursively).
→ `text-analysis`	Textual content and its analysis. It applies a set of pattern matching expressions and keyword extraction to the extracted text.
`Use-Case$Folder`	Folder of the analysed file, as a means to identify a "use-case" in a data set.
`Associated-Metadata`	Structured data associated to the file extracted from an associated file, such as an Excel spreadsheet. The objective is to include data from a companion table attached to the analysed file. *MEL* analyses the attached file to the data set in order to find and extract the associated data fields for the current processing file.
`_attachments`	Only when storing the results in the document store. Holds the complete file content as text (if the file is binary, a Base64 encoding is performed).
`NLP-NER`	Only when *TNNT* is enabled. For each analysed document (compressed `.zip` files and `.msg` e-mails may contain several documents or attachments), the results of the NER analysis are generated per enabled model and recognised category.
`NLP-NER-Summary`	Only when *TNNT* is enabled. A bottom-up summary of all recognised entities from the NER analysis (per document per entity). For each entity, a frequency (how many times the entity was recognised) per model and category is calculated.

the ground for content-based analysis. *MEL* is integrated with "The NLP[7]-NER[8] Toolkit" (*TNNT*), which automates the extraction task of categorised named entities from the *MEL* results by using diverse state-of-the-art NLP tools and NER models [5]. *MEL* implements primitives for metadata and content extraction from unstructured data sets of heterogeneous formats, and along with the *TNNT* results, it provides the groundwork for content-based analysis. *MEL* and *TNNT* were developed in conjunction with *J2RM* [4], to easily map the JSON results to RDF as part of an automated KGCP[9].

2 Core Features

MEL has comprehensive metadata extraction support of various file types and formats. In a nutshell: (1) it takes as input a document (file) set; (2) then, for each document, it extracts its related metadata and content-based information, while performing basic text analysis (such as applying a configurable set of regular expressions and keyword extraction task); and, (3) as output, it generates a JSON file with the extracted metadata and text content with a structure based on the supported formats' document object model. It can store the results in a document store[10]. *MEL*'s general output structure is presented in Table 1. *MEL* has a detailed configuration JSON file that defines how the processing will be performed through a set of parameters and flags that establish the initial settings related to the document store, input document sets, *TNNT* general configuration, file extension mappings, the "`Associated-Metadata`" processing (Table 1), and regular expressions to apply in the text analysis task, among other

[7] Natural Language Processing
[8] Name Entity Recognition
[9] `https://w3id.org/kgcp`
[10] Currently, the tool only supports **CouchDB** (`https://couchdb.apache.org/`).

Table 2. *MEL* supported file types

File Type	Description (General-Metadata: **17 to 27 attributes**; Specific-Metadata: **for each format**)	Max. Number	Avg. Number
.pdf	the tool uses the **textract Tesseract-OCR** method[a] and **pdftotext**[b] tools.	55	24
.docx	Microsoft Office "core properties" extraction and document model: headings*, text-para+, bold*, italic*, underline*, tables*, hyperlinks*, sections+).	>50	28
.pptx	Microsoft Office "core properties" extraction and document model: slides+ (placeholders+: tables*, images*, text*, notes*, etc.), bold*, italic*, underline*, hyperlinks*).	>60	34
.doc, .xls, .ppt, .vsd, .mpp	These formats are generalised to OLE 2. OLE 2 formats comprise of over 30 properties and multiple "streams and storages" components.	>60	45
.msg	Uses MAPI[c] and OLE 2 format extraction; also, it follows a recursive processing model for e-mail attachments. Document model: e-mail properties+ (subject, body, HTML-body, etc), To*, CC*, BCC*, and attachment property set	>47	54
.docm	Uses a C# (.NET v4.8) converter to .doc (it removes the macros).	>60	45
.xlsx	Extracts the table structure of each Workbook. Values are encoded as strings.	—	—
.csv	Extracts the table structure by rows and by columns (transposed).	—	—
.rtf	Extracts the full rich text along with tags.	—	—
.txt	(.xml, .html, .htm, .json) All processed as raw text files. For JSON files, a copy of the full structure is kept in "Associated-Metadata".	—	—
.zip	Follows a recursive processing model for each file in the compressed ZIP.	—	—
Image formats	Various formats such as .jpg and .png.	>27	19

[a] https://textract.readthedocs.io/, [b] https://www.xpdfreader.com/, [c] https://docs.microsoft.com/en-us/office/client-developer/outlook/mapi/outlook-mapi-reference

settings. The supported file types are presented in Table 2. The third column shows the theoretical number of attributes that the tool is able to extract per document type, whilst the fourth column shows the average of the extracted attributes from four use case document sets[11]. OLE 2 file types[12] and `.docm` can only be processed on Windows operating systems. Specifically for OLE 2 file types, *MEL* uses the **olemeta** tool[13].

3 Architecture

MEL is fully integrated with *TNNT* as depicted in Figure 1. The set of Python-based methods implemented in *MEL* are generic and can be applied to extract the content and metadata of all supported file types. *MEL* uses various open-source packages and tools with complementary capabilities to form a "Swiss army knife" of metadata and content-based information extraction from heterogeneous document sets. As part of the "`General-Metadata`" extraction task, *MEL* optionally uses the XML[14] output from the **NLNZ Metadata Extractor** tool[15], a Java standalone tool that extracts a comprehensive attribute and

[11] Each set ranges from 1,174 to 3,334 documents. They are government documents about assessments, endangered species, federal budget, and procurement, all from the "Australian Government Records Interoperability Framework" (AGRIF) [1] project.

[12] https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-oleds/fdc5e702-d09e-4344-a77f-eb079d41f23f

[13] http://www.decalage.info/python/oletools/

[14] Extensible Markup Language.

[15] http://meta-extractor.sourceforge.net/

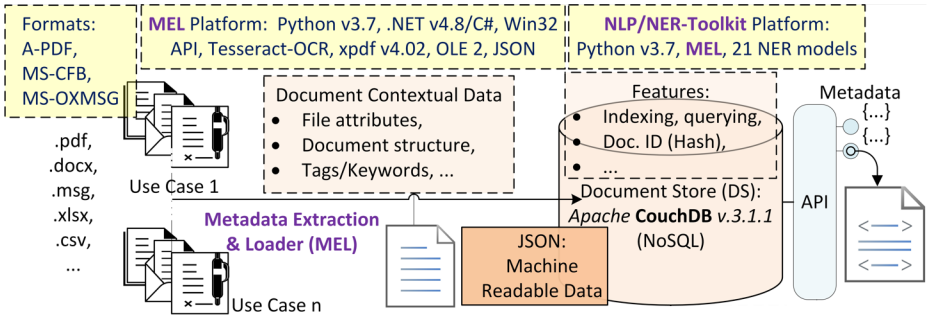

Fig. 1. *MEL* overview as part of a KGCP

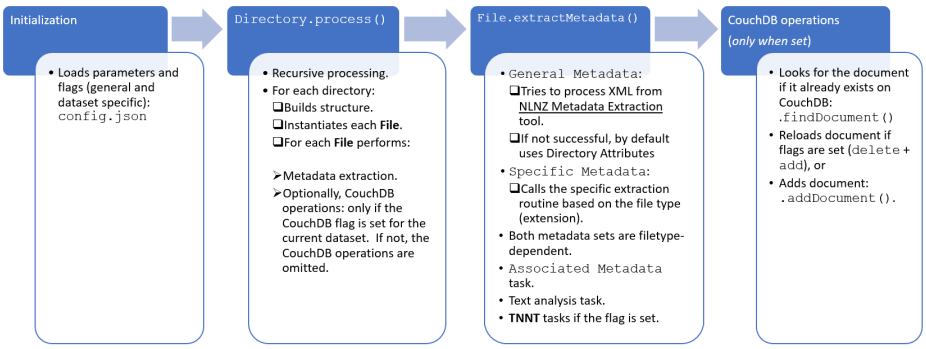

Fig. 2. Processing Model

property list from dozens of file formats. The *MEL* general processing model is presented in Figure 2. It is important to note that each file type has its own specific processing model as well as the text analysis task, which is the last step that is performed for any output.

4 Related Work

The most comprehensive and current state-of-the-art tool for content extraction and analysis is *Apache Tika*[16], which is a complete and complex Java-based general-purpose system. While *MEL*'s core goals resemble the ones of *Apache Tika*, the main difference and benefit of *MEL* as compared to *Apache Tika* is that it is a lightweight Python-based package for the metadata extraction of common file formats aimed to be used in a KGCP. Although there is a wide range of Python-based tools and libraries for metadata extraction, to the best of our knowledge, there is no package available that fully integrates in one system a comprehensive set of methods for metadata and content extraction of common file formats that generate the results in JSON structures based on the document

[16] `https://tika.apache.org/`

object model of each format type. Last, *MEL* can assist in the information extraction stage of several KGCPs, such as the ones described in [6], [2], and [3].

5 Conclusions and Future Work

MEL provides a versatile mechanism to extract metadata and content-based information from unstructured data sets of heterogeneous file formats, agnostic of the data sets' domain (general purpose). It has been tested over thousands of documents using different formats and datasets as part of the AGRIF project. Based on the structure of the *MEL*'s JSON results, it is possible to easily add a vocabulary or light-weight ontology using JSON-LD[17] annotations, in order to make the extracted metadata "RDF ready". This will be explored in the near future leveraging on the integration with JSON-LD ontologies. More file formats will be added in a per use-case requirements basis, in order to support KGCP tasks. Additionally, a project to "containerise" the *MEL+TNNT* tools is planned in the near future.

The major contributions of this tool are: (1) the ability to extract metadata sets and content-based information from different source document formats; (2) the comprehensive support of over 20 different file types/formats integrated into one easy-to-use Python-based system; (3) integration with *TNNT* which automates the extraction of categorised named entities from the results by using diverse state-of-the-art NLP tools and NER models; and (4) the JSON result files can be easily mapped to RDF using *J2RM*.

References

1. Department of Finance: AGRIF Ontology. Ontology Specification, Australian Government (2018), `http://linked.data.gov.au/def/agrif`
2. Elhammadi, S., V.S. Lakshmanan, L., Ng, R., Simpson, M., Huai, B., Wang, Z., Wang, L.: A high precision pipeline for financial knowledge graph construction. In: Proceedings of the 28th International Conference on Computational Linguistics. pp. 967–977 (2020). https://doi.org/10.18653/v1/2020.coling-main.84
3. Jia, Y., Liu, D., Sheng, Z., Feng, L., Liu, Y., Guo, S.: EasyKG: An End-to-End Knowledge Graph Construction System, pp. 221–228 (2020). https://doi.org/10.1007/978-981-15-3412-6_22
4. Rodríguez Méndez, S.J., Haller, A., Omran, P.G., Taylor, K.: J2RM: an Ontology-based JSON-to-RDF Mapping Tool. In: ISWC: Posters & Demos Track. vol. 2721, pp. 368–373. CEUR (2020), `http://ceur-ws.org/Vol-2721/paper593.pdf`
5. Seneviratne, S., Rodríguez Méndez, S.J., Zhang, X., Omran, P.G., Taylor, K., Haller, A.: TNNT: The Named Entity Recognition Toolkit. In: arXiv (2021), `https://arxiv.org/abs/2108.13700`
6. Simsek, U., Umbrich, J., Fensel, D.: Towards a knowledge graph lifecycle: A pipeline for the population of a commercial knowledge graph. In: Qurator (2020), `http://ceur-ws.org/Vol-2535/paper_10.pdf`

[17] `https://www.w3.org/TR/json-ld11/`