Corpona – The Pythonic Way of Processing Corpora

Khalid Alnajjar[0000−0002−7986−2994] and
Mika Hämäläinen[0000−0001−9315−1278]

Department of Digital Humanities, University of Helsinki
firstname.lastname@helsinki.fi

Abstract. Every NLP researcher has to work with different XML or JSON encoded files. This often involves writing code that serves a very specific purpose. Corpona is meant to streamline any workflow that involves XML and JSON based corpora, by offering easy and reusable functionalities. The current functionalities relate to easy parsing and access to XML files, easy access to sub-items in a nested JSON structure and visualization of a complex data structure. Corpona is fully open-source and it is available on GitHub and Zenodo.

Keywords: XML data · corpus processing · open source

1 Introduction

In the era of machine learning, corpora have become one of the most important resources for NLP research. However, there is no one standard for annotating data or representing existing linguistic data, in fact there are several of them: Giella's XMLs [5], TEI (see [3]), CoNLL-U [17], ELAN XML [16], EXMARaLDA XML [12] and NewsML XML [1] to name a few. There is such a variety of different ways of representing data that an NLP researcher is bound to spend a whole lot of time in converting them from one format to another. For this purpose, we have implemented Corpona[1]. While several other corpus processing tools exist [13, 14, 6], we aim for simplicity and reusability with Corpona.

Corpona is an open source Python library licensed under the Apache-2.0 License. Each release version is uploaded and permanently archived automatically to Zenodo. The library is easy to install through pip (pip install corpona).

While working with XML data in various projects such as Ve'rdd [2] and neologism retrieval [11], we have found ourselves writing similar parsing code for a variety of different tasks. This called for a more centralized approach where code reuse can be maximized. This need gave the initial idea for Corpona, a one-stop tool for XML and JSON dataset processing. We needed a fast way of getting things done with as little new code as possible. As some of the XML structures we have worked with, such as Giella XML, are in use in multiple tools such as click-in-text dictionaries [7] and online learning tools [15], the features implemented in Corpona are potentially useful for a wider audience.

[1] https://github.com/mokha/corpona

In Hämäläinen, M., Partanen, N., Alnajjar, K. (eds.) Multilingual Facilitation (2021), pages 25–30.

2 The Current Architecture

At the current stage, Corpona consists of three main modules: *xml*, *summary* and *explorer*. The main functionalities of these individual modules are easily accessible from the main *corpona* module.

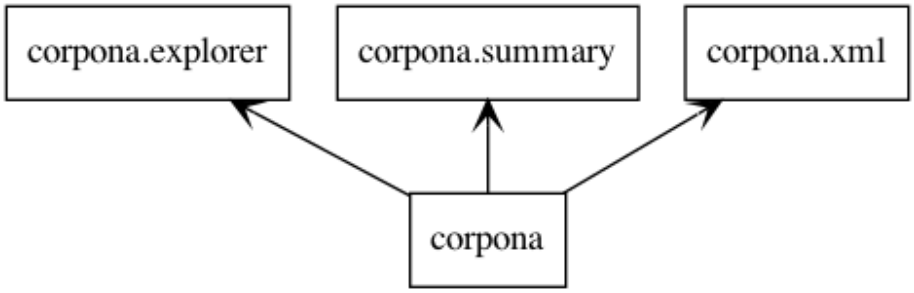

Fig. 1. A UML diagram showing the modules in corpona.

The modular structure is seen in Figure 1. This structure has been crafted keeping in mind the future development directions of the library. Only the *xml* module has classes. The class diagram is shown in Figure 2.

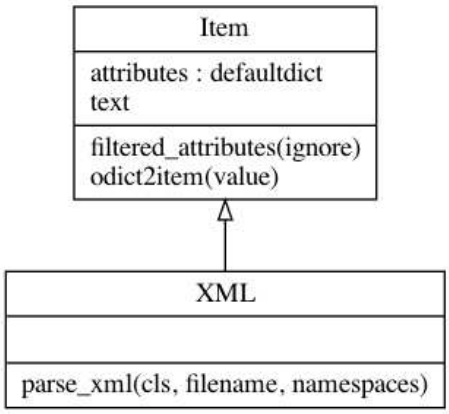

Fig. 2. A UML diagram showing the classes in the *xml* module

3 Functionalities

In this section, we will describe the main features of Corpona. We go through every module and show usage examples to illustrate their use with real world data.

```
from corpona import find_child

data = {"key":["list_item", {"key2":"oo"}, {"key2":"bbb"}]}
print(find_child(data, ["key", "key2"]))
print(find_child(data, ["key", "key3"], default_value="ok"))

>> ['oo', 'bbb']
>> ['ok']
```

Fig. 3. An example of the *find_child* method in the *explorer* module

Figure 3 shows the main functionality of the *explorer* module. The *find_child* method is useful for getting the data recorded in a sub-dictionary. The method takes in a dictionary and a path consisting of keys. The method will automatically go into lists and loop through their sub-elements finding dictionaries with keys indicated in the path. We find this feature useful as usually we just want certain data out from a JSON or an XML based on dictionary keys regardless of whether they were inside of a list or not. The method also takes an optional *default_value* parameter. This value will be returned in case no item matched the query. As the method automatically loops through lists, it is possible that there are multiple dictionaries that meet the criteria set in the path, for this reason the method always outputs a list.

Figure 4 shows how to use the *summarize* method from the *summary* module. The method takes in a complex dictionary structure loaded from a JSON file or parsed from an XML using Corpona. The method produces a quick overview of the structure of the data as seen in the example output. It is a fast way of seeing what keys are in the dictionary and what the data types are that are stored under each key. This is very useful for better understanding the structure of a new dataset one starts to work with.

Figure 5 shows the XML parser in action. The parser takes in a path to a file, and parses it into a manageable *Item* structure. Corpona makes it possible to loop through the different parts of the XML in an easy fashion. The sub-elements can be looped by getting an item by tag name from the *Item* class. The XML attributes can be accessed directly e.g. *d.href* would return the *href* attribute of the *Item* object *d*.

```
from corpona import summarize
from pprint import pprint

pprint(summarize([
    {'key1': 'hello1', 'key2': 1},
    {'key1': 'hello2', 'key2': 2},
    {'key1': 'hello3', 'key2': 3},
    {'key1': 'hello4', 'key2': 4},
]), indent=4)

>> [{'dict': [{'key1': ['str'], 'key2': ['int']}]}]
```

Fig. 4. An example of the *summarize* method in the *summary* module

```
from corpona import XML
d = XML.parse_xml("test.xml")
for item in d['p']:
    print(item)
>> Hello
>> World
```

Fig. 5. An example of the *XML* class in the *xml* module

4 Conclusions and Future Features

In this paper, we have presented Corpona, an open-source Python library for corpus processing. We have described the main functionalities and demonstrated their use with examples.

Conversion between different formats is on the long-term road-map of this library. Some existing approaches such as converting Giella XMLs to TEI format [8] could be incorporated in the future. This not only makes the data more accessible in Python but also facilitates its reuse on different platforms that operate on different XML structures.

We are also thinking of different ways of visualizing data. The biggest challenge when you are given a dataset, is to know exactly what it has, what the structure is, what elements can contain lists, strings, numbers, null types etc. Having a simple way of visualizing the structure helps in understanding how to approach the data. The current implementation in the *summary* module is a good start but it might still produce an overly complex output for large and inconsistent dictionaries.

Despite the growing number of Universal Dependencies annotated corpora for endangered Uralic languages [10, 9], we do not currently have any plans to in-

corporate a CoNNL-U parser in Corpona as this feature is available in our other library called UralicNLP [4]. However, UralicNLP does provide rudimentary access to Giella XML dictionaries. In the future, Corpona should be included in UralicNLP as a dependency for better parsing these files.

References

1. Allday, T.: Newsml — enabling a standards-led revolution in news publishing? In: XML TECHNOLOGIES IN BROADCASTING (1998)
2. Alnajjar, K., Hämäläinen, M., Rueter, J., Partanen, N.: Ve'rdd. narrowing the gap between paper dictionaries, low-resource nlp and community involvement. In: Proceedings of the 28th International Conference on Computational Linguistics: System Demonstrations. pp. 1–6 (2020)
3. Bański, P., Bowers, J., Erjavec, T.: Tei-lex0 guidelines for the encoding of dictionary information on written and spoken forms. In: Electronic Lexicography in the 21st Century: Proceedings of eLex 2017 Conference (2017)
4. Hämäläinen, M.: UralicNLP: An NLP library for Uralic languages. Journal of Open Source Software **4**(37), 1345 (2019). https://doi.org/10.21105/joss.01345
5. Moshagen, S., Rueter, J., Pirinen, T., Trosterud, T., Tyers, F.M.: Open-Source Infrastructures for Collaborative Work on Under-Resourced Languages (2014), the LREC 2014 Workshop "CCURL 2014 - Collaboration and Computing for Under-Resourced Languages in the Linked Open Data Era"
6. Rayson, P.: Wmatrix: a web-based corpus processing environment. In: Citeseer (2009)
7. Rueter, J.: Giellatekno open-source click-in-text dictionaries for bringing closely related languages into contact. In: Proceedings of the Third Workshop on Computational Linguistics for Uralic Languages. pp. 8–9 (2017)
8. Rueter, J., Hämäläinen, M.: On xml-mediawiki resources, endangered languages and tei compatibility, multilingual dictionaries for endangered languages. In: Gürlek, M., Çiçekler, A., Taşdemir, Y. (eds.) AsiaLex 2019. Asos Publisher, Turkey (2019)
9. Rueter, J., Partanen, N., Ponomareva, L.: On the questions in developing computational infrastructure for komi-permyak. In: Proceedings of the Sixth International Workshop on Computational Linguistics of Uralic Languages. pp. 15–25 (2020)
10. Rueter, J.M., Tyers, F.M.: Towards an open-source universal-dependency treebank for erzya. In: International Workshop for Computational Linguistics of Uralic Languages (2018)
11. Säily, T., Mäkelä, E., Hämäläinen, M.: Explorations into the social contexts of neologism use in early english correspondence. Pragmatics & Cognition **25**(1), 30–49 (2018). https://doi.org/10.1075/pc.18001.sai
12. Schmidt, T., Wörner, K.: Exmaralda. In: The Oxford handbook of corpus phonology (2014)
13. Silberztein, M.: Intex: a corpus processing system. In: COLING 1994 Volume 1: The 15th International Conference on Computational Linguistics (1994)
14. Silberztein, M.: Nooj: a linguistic annotation system for corpus processing. In: Proceedings of HLT/EMNLP 2005 Interactive Demonstrations. pp. 10–11 (2005)
15. Uibo, H., Rueter, J., Iva, S.: Building and using language resources and infrastructure to develop e-learning programs for a minority language. In: Proceedings of the joint workshop on NLP for Computer Assisted Language Learning and NLP for Language Acquisition. pp. 61–67 (2017)

In Hämäläinen, M., Partanen, N., Alnajjar, K. (eds.) Multilingual Facilitation (2021), pages 25–30.

16. Wittenburg, P., Brugman, H., Russel, A., Klassmann, A., Sloetjes, H.: Elan: a professional framework for multimodality research. In: 5th International Conference on Language Resources and Evaluation (LREC 2006). pp. 1556–1559 (2006)
17. Zeman, D., Nivre, J., Abrams, M., Ackermann, E., Aepli, N., Aghaei, H., Agić, Ž., Ahmadi, A., Ahrenberg, et al.: Universal dependencies 2.7 (2020), http://hdl.handle.net/11234/1-3424, LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University