Designing and programming converter scripts for ThermoML

Andrey Belkin

May 6, 2021

Institute of Biochemistry and Technical Biochemistry - University of Stuttgart

Contents

1 Task

Experimental and simulated data of solvents are often used to evaluate certain behaviour in biochemistry. This data is usually found in figures, plots and tables of scientific papers. However there is a lack of an easy accessible data format to quickly retrieve, compare or write solvent data. The task of this internship was to design a script to enhance the reading and writing process of solvent data. The foundation was the already established, yet rarely implemented format ThermoML.

2 Practical Work

2.1 ThermoML Data structure

ThermoML is an XML based markup language to store scientific measurements. A list of nested tags describe different parts of the measurement. A more detailed documentation will be provided as an appendix to the report. It should be noted however, that even more distinct tags are available, which go beyond the scope of this task. Different mappings are available, since the format has been updated multiple times. The 2010 version seems to be the best documented so far, despite a 2017 version being in construction.[1] Unfortunately as of 16. Apr 2021 there is no code available on GitHub.

Previous iplementations as a database are deprecated and the server has been shut down. [2]. As of 16. Apr 2021 the code is available via Github or the conda channel choderalab. The code attempts to connect to a server to pull the necessary data for the pandas dataframe, however the server is not available and replies with an error. Unfortunately it was therefore not possible to implement thermopyl in this work.
ThermoML is roughly divided into three parts:

2.1.1 Citation

The citation contains all necessary information linking the ThermoML file to the adjectant paper. Most importantly the authors, year of publication and the DOI is noted. The title of the paper and the abstract can be provided as well. Furthermore information on the paper (journal, volume, page) can be specified.

```
<eType>journal</eType>
<eSourceType>Original</eSourceType>
<sAuthor>Yadav, A.[Anita]</sAuthor>
<sAuthor>Pandey, S.[Siddharth]</sAuthor>
<sPubName>J. Chem. Eng. Data</sPubName>
<yrPubYr>2014</yrPubYr>
<dateCit>2020-09-29</dateCit>
<sTitle>Densities and Viscosities of (Choline Chloride + Urea)
<sAbstract>Deep eutectic solvents (DESs) have been regarded as
<sDOI>10.1021/je5001796</sDOI>
<sIDNum>7</sIDNum>
<sVol>59</sVol>
<sPage>2221-2229</sPage>
```

Figure 1: Screenshot of the Citation structure of a ThermoML file.

[1] https://github.com/usnistgov/thermoML
[2] https://github.com/choderalab/thermopyl

2.1.2 Compounds

The second part of the ThermoML file is the compound declaration. Each compound that will be referenced in the ThermoML file later has to be declared as a compound first. The compound is provided with a unique identifier number (usually starting with 1) to reference it later. It can be furthermore described by its chemical structure, its common name, a smiles code or a International Chemical Identifier. The ionic charge can be specified if needed. Each compound can have an arbitrary amount of samples. These are also enumerated. Each sample has a purity and an analysis method assigned to it.

2.2 Measured Data

The biggest part of the file is the Measured Data. Contrary to other formats it is possible to store as many distinct measurements as required. Each measurement is initialised by declaring, which compounds (as described in 2.1.2) and their respective samples are used. Followed by that a detailed description of the measured properties is provided. An arbitrary amount of properties can be stored. Each property is also assigned with a unique identification number. Furhtermore more descriptions are provided, such as method of measurement, unit and the phase in which the measurement took place.

After that the variables are initialised. In the scope of this task there were only three possible variable types: Temperature, mole fraction and the respective measured property as decribed above. Thereafter follows an arbitrary amount of datapoints which describes all the measured variables and their respective error values.

2.3 Script design

For the purpose of this task two scripts were designed. The first converts a set of csv templated into a complete ThermoML file while the second converts all available info from a chemical markup language (CML) file into an incomplete ThermoML file. The scripts are available as part of an open source repository on GitHub: `https://github.com/Sentexi/thermoml_converter`

2.3.1 csv2thermoml

Since ThermoML is a markup language a script had to be designed to make reading and writing it as easy as poissible. Since a ThermoML reader already exists[3] the project at hand was mainly concerned with writing ThermoML files. The focus was set on two points: On the one hand the implementation should be as easy and recognisible as possible, on the other hand it should be user friendly, considering that many people who conduct experiments do not want to be concerned with programming or different file types or formats. As such a set of documented CSV templates was chosen. For each file

[3] `https://trc.nist.gov/ThermoML_Opener.html`

one citation file with a quick documentation in the first column was provided. Furthermore each compound and each dataset had an own CSV template. The templates are available in the GitHub repository as well.

The script design itself was to create the tree of nested tags first and fill in the results later. For this a function `create_xml_subelement_with_list` was created, that distinguishes between tags with an entry and tags that only are parents to other tags. The content of the csv was read in as a numpy array and flattened in cases, where more than one entry per tag was entered. In the end a loop ran over all created tag elements which were marked to fill with content and the list obtained from the csv. The combined xml parts were added together and stored in an xml document.

In all cases some preprocessing was necessary. In case of the citation the three letter codes of the authors were generated and the position of the author tag was shifted. For compounds and data points the flattening of the lists led to some rearrangement in the positions of the values.

2.3.2 cml2thermoml

This script was designed to transfer data stored as CML file into ThermoML files. However, ThermoML provides many more options than CML and so only all CML values are transferred, leaving a partly unfinished ThermoML file. For this the CML file is read in as an xml document and a loop cycles over all nested tags, assigning values to a list and transcribing them to the proper tags for ThermoML. Most functionalities for this were taken from the csv script (2.3.1).

For this reason it is only a minor task to add the missing values from a csv template. This means that a combination of CML files and csv templates can be used to combine multiple cml files into one ThermoML file. However to keep the overview easy templates and CML files were not entangled in the scope of this work.

3 Repository organisation

The repository is subdivided into six folders for better usefulness. In the folder "CSV" there are the templates necessary for the construction of a ThermoML file. Each ThermoML file takes exactly one Citation template and as many Compound and Data templates as required. They have to be labeled in such a way, that their respective purpose is written in the beginning (e.g. Citation, Compound or Data [case sensitive!]) and a number at the end.

A filled out set of templates can then be put in an arbitrarily named folder and put into the folder "src". The csv2thermoml script transforms all sets of templates found in src. A prefilled example is provided in the folder "Example".

For transcription of CML files the CML file has to be sepecified in line 218 of the cml2thermoml script.

All ThermoML files are stored in the folder output, using the first three letters of the first authors and the publication year as name. If two file names are equal, the previous file is overwritten.

For further development a ThermoML mapping from 2010 to 2017 is provided as well. In it a variety of tags can be found to enhance the scripts in the future.

4 Next steps

As stated previously the project could not be finished in the scope of a six week internship. Thus naturally some work still remains to be done in order to finalise the script. The csv2thermoml script described in (2.3.1) is finished in that sense that from a set of csv templates a ThermoML file can be constructed. However in future works it is advisable to write a function that takes other objects as well, such as pandas dataframes or numpy arrays.

Since ThermoML has a vast majority of different tags it would be useful to customise the script to autodetect different units and physical values and automate the implementation even further. Also more output formats and an automatic analysis of certain value with autogeneration of plots are useful.

The csml2thermoml script (2.3.2) extracts all available data from a cml file. However a ThermoML file provides much bigger variety of tags to give more information about the measurement and simulation. A csv template or other ways have yet to be found to fill in the missing data from the CML file.

5 Outlook

ThermoML is an ambitious attempt at standardising all available data for solvents and make them comparable on a broader scale. The main aspects of it's success are usability and applicability. For the first part the creation of an API would be advisable. This would give the opportunity to connect simulation workflows directly to a standardised output. On the other hand a proper ThermoML reader could perform a lot of standard analysis with such files in bulk and make them storable and comparable. A database of ThermoML files would make the search for experimental data way easier than it is now.

This also gives the opportunity to construct a web app that combines input, database search and ouput in a way that is maximum user friendly and requires no programming knowledge whatsoever. This would greatly benefit the application of the format.

It would also open the field for mass data processing: Be it a meta study of simulated data vs. experimental results or machine learning, processing big loads of data is currently near impossible without hours of aggregating data or resorting to potentially expensive commercial alternatives. ThermoML could be the beginning of an open source database for scientific entries to make experimental and simulated data more accessible globally.