Lexicons of Early Modern English (LEME): Work Study Project Report

By Victoria R. Spada

Published: February 19, 2021.

This report provides a summary of the work completed by Victoria Spada for Job ID 167141 (LEME IT Research Assistant) during the 2020-2021 academic year.

The intent of this report is to describe the project(s) I have worked on, how much has been done, what need to do be done, and a rough estimate of the hours required.

Transcribing a Text

The first task that I took on was transcribing a French-English text from 1521 by Alexander Barclay. This task was done to familiarize myself with LEME's encoding rules. The finished transcribed text was saved in **L:\Victoria\Barclay encoding text** as a .docx and .txt file ('BarclayComplete.docx' and 'BarclayComplete.txt'), as well as a processed .xml file ('BarclayComplete.txt.prep.xml').

In February 2021, Prof. Lancashire went over my transcription and modified it to better apply LEME's encoding rules. We both contributed to resolving any errors using my validation program, until no errors remained. This modified version is saved in the same folders as the others, as '705Barclay1521VictoriaIanPlainText.txt' and '705Barclay1521VictoriaIanPlainText_prep.xml'.

The LEME Validation Program

The aim for the validation program was to have an application that runs on Windows 10 that can be used to validate the encoding of XML-encoded documents.

The preprocessing and validation programs were created to:

1. take an input .txt file and output the same file as an .xml file.
2. scan the outputted .xml file from the preprocessing program and validate whether the file:
    a. is written as well-formed XML.
    b. is written according to LEME's encoding rules.

These programs were both written in python (version 3.7) and were turned into executables for usability.

The preprocessing program accepts a .txt file containing an encoded text, and converts it to an .xml document. The main validation program then scans for syntactical errors and schema-related errors (ways in which the given document goes against LEME's encoding rules) and reports the location of these errors.

After running the main program, if there are any errors, the user must use the saved error logs as a guide to resolve the issues. Debugging the .xml file may be iterative process, and the program displays a positive message when the .xml file is error-free.

PREPROCESSING PROGRAM

CREDIT*:* The code for this program was taken from Tim Alberdingk Thijm's (Princeton University) work in L:\SchemaValidation, and slightly edited by Victoria Spada in 2021.

Files: leme.dtd, txt_to_xml.exe

The executable is located in **L:\Victoria\XML_Validation\dist\.**

Functions

This program performs the preprocessing steps to fix .txt documents and create well-formed XML. The .txt file being converted must in the directory: **L:\Victoria\XML_Validation\Text and XML Files**.

Upon opening the executable, the words "What is the name of the .txt file to be converted to an .xml file?" will appear. The name of the file is then typed in as an input (ie, 'test.txt'), and the XML version of the file is saved in the directory where the text file is located (ie, as 'test_prep.xml'). The user can then press 'Enter' to exit the application, or type any other key to convert another .txt file to .xml.

The program uses functions written by Thijm to solve common grammar errors, close certain unclosed brackets, and add a header to the file. The file is then saved as an .xml file with the suffix "_prep" added to the end of the file name. The file is then saved in the directory where the .txt file is.

Note that this program does not work to validate that the XML is well-formed or follows LEME's encoding rules: those are the functions of the validation program.

VALIDATION PROGRAM

Files: leme.xsd, main.exe, main.py, validator.py

The executable is located in **L:\Victoria\XML_Validation\dist\.**

Functions

Upon opening the executable, the message "What is the name of the file to be validated?" is displayed. The program takes the name of an .xml file as an input entered by the user (ie, test_prep.xml) and validates it. The .xml file being validated must in the directory: **L:\Victoria\XML_Validation\Text and XML Files**. The file being validated should be an output .xml file from using the preprocessing program.

The program takes the name of an .xml file as an input and validates it by calling validator() in validator.py, which validates the file against XML syntax rules (criterion "a") and the rules outlined in the XML Schema Definition (XSD) leme.xsd (criterion "b"). The program prints a message stating whether the file is valid.

The program checks that the file meets criteria "a" and "b" outlined above. If criterion "a" is met, the message "XML well formed, syntax ok." is printed. If criterion "b" is met, then the message "Encoding valid, schema validation ok." is printed.

Error messages regarding whether the XML is well-formed (criterion "a") are saved into a file called "error_syntax.log", and errors with respect to LEME's encoding rules (criterion "b") are reported in "error_schema.log". These files can be found in **L:\Victoria\XML_Validation\Error Logs** and can be read using Notepad. The user should read these logs after exiting the application to resolve these issues in the .xml file, and then re-run the validation program on the updated file to check that no issues remain.

NOTE: The program is incapable of checking criterion “b” until criterion “a” has been met. This is the way that the program currently works, and the computer will not read the XML until it is well-formed. If there are both syntactical and schematic errors, until one resolves the syntactical errors, the program will not output a ‘error_schema.log’ file. Once the syntactical errors have been resolved, re-run the improved .xml file through the program and check if there are any schema-related errors.

Feedback and Next Steps

The program was tested by Prof. Lancashire in February 2021.

The initial goals of the project were that the two programs do the following:

1. The first validates the tags in a text so that it can be loaded into the online database.
2. The second validates an XML-encoded file that is generated by a standalone program from that previously validated text.

The approach that I took is differs from what was proposed here and validates the XML before validating the LEME encoding. This order of validation makes more sense with the python package lxml that the program depends on (lxml.etree in particular), which can only scan for valid tags against a user-defined schema if the XML is well-formed.

Based on user feedback from LEME’s researchers, I would like to qualitatively evaluate the usability of the program and see how it can be improved on and expanded to help LEME reach its goals. Depending on the feedback received, the timeline of completing the program can vary. I am interested in answers to the following questions:

1. How user-friendly would you describe my validation program to be?
2. What functionalities would you like to be added to the program (specific or broad ideas)?

From there I can add new modules to the program.