**Word Data Extractor User Guide**

Hello, and thanks for using our Word Data Extractor! This guide should help you get started as well as understand how to use our software.

**Getting Started**

The Word Data Extractor has a few system requirements that must be met for its use:

1. Python 3.3 or newer is installed.
2. The Python-docx module is installed in the version of Python you will be using.
3. The Docx2txt module is installed in the version of Python you will be using.
4. The Docx2python module is installed in the version of Python you will be using.

Once you ensure that all those requirements are met you can download our source code from \*Insert Repository Here\* you need to ensure that you save all the source code files in the same folder as they depend on each other. Once you’ve done that you should be ready to run the Word Data Extractor.

**Executing the Word Data Extractor**

While there are many ways you can execute python scripts, this guide will focus on running the Word Data Extractor from Windows Command Prompt. You can follow the following steps to run the Word Data Extractor.

1. Open a Windows Command Prompt terminal.
2. Change your active directory to the directory where you have the Word Data Extractor source code files. This is done by using the “cd” command followed by the path to the directory of the Word Data Extractor source code.
3. Execute the “WordDataExtractor.py” script using the appropriate Python interpreter. Ensure that the interpreter you use meets the requirements in the “Getting Started” section. An example command that you could use for this is “py -3.10 WordDataExtractor.py” if you wanted to use a Python 3.10 interpreter to execute the Word Data Extractor.

**Using the Word Data Extractor**

Once you’ve executed the Word Data Extractor, you’ll see a menu displayed with the basic commands that you can issue. These commands are “Outname”, “Outloc”, “Input”, “Parse”, “Help”, and “Exit”. More information about each of these commands is shown below:

Outname: This command will allow the user to specify the name for the saved XML file. Typing outname will result in the program prompting the user to type in a specific, valid file name that the XML will be saved as.

Outloc: This command will allow the user to specify the location in which the XML will be saved. Typing outloc will result in the program prompting the user to type in a specific, valid folder location in their system for the XML to be saved within.

Input: This command will allow the user to specify the .docx file to be parsed. Typing input will result in the program prompting the user to enter a specific, valid filepath to the .docx file to be parsed.

Parse: This command will parse the specified .docx file. Typing parse will result in the program prompting you for details about procedures in the .docx file to be parsed and then running through its parse script, parsing the .docx file into its XML equivalent.

Help: This command will provide information about commands that you can execute in the Word Data Extractor. The following are help commands; help, help input, help parse, help outname, help outloc, and help exit.

Exit: This command will return the user to a previous menu or stop the program depending on if it’s used in the help menu or in the main menu.

**Difficulty Extracting Procedures**

There are also 2 additional commands that you can execute if you are having difficulty parsing procedures. They are “View”, and “Patterns” More information about them below:

View: The view command allows you to see the 4-dimensional table that is output by the docx2python module from the .docx file that it receives. This could allow you to identify the exact string patterns that surround the tokens that help the parser identify procedures, in case they aren’t what is pre-programmed in. (What’s pre-programmed is “” as procedure pre-pattern, “)\t” as procedure post-pattern, “\t” as sub-procedure pre-pattern, and “)\t” as sub-procedure post-pattern.)

Patterns: The patterns command allows you to set the pre-pattern and post-patterns for tokens that the parser will use to identify procedures. As an example, you might have a procedure that can be identified by the numeric tokens (1, 2, 3…) but the procedure as it appears in your word document has these tokens preceded by 4 spaces instead of a tab character. You could use the Patterns command to change the procedure pre-pattern to 4 spaces. One thing to note is that if your patterns contain tabs (“\t”) use the tab key rather than typing “\t” to enter them properly.

**Known Defects and Issues**

There are a few known defects and issues with the Word Data Extractor, they are listed here:

1. Characters not included by the UTF-8 character set cannot be included in the word document. If they are, generating the XML will crash the program.
2. Hyperlinks and Textboxes cannot be included in the word document. The Python-docx module which is used extensively in the project cannot see them, and including them will cause a desync of the internal document structure and the 4-dimensional table from the docx2python module, potentially leading to incorrect XML generation or a crash.
3. The current version of Word Data Extractor doesn’t support sub-procedures within tables, so it’s advisable to not try and extract documents that contain those.