NLP - Exercise 1

Student ID: 211406343

Part 1:

1. It has a very structed format for every paragraph, sentence and token along its annotation and metadata.

2. The richness of the content is limitless.

3. Advantage:

   The data stored using XML can be changed at any point of time without affecting the data presentation.

   Disadvantages:

   a. The syntax is verbose and redundant.

   b. The document is less readable compared to other formats such as JSON.

Part 2:

For the Token class, I decided to declare 5 attributes:

1. t: Stands for token type.
2. word: The word itself as a string.
3. c5: Represents the grammatical class.
4. hw: The same word but in lowercase (equal word).
5. pos: Part of speech.

For the Sentence class, I declared 3 attributes:

1. tokens: A list of Token objects containing all the tokens in the sentence.
2. parent: Decides whether the sentence is a header or in a paragraph.
3. size: The number of tokens in the sentence.

And finally, for the Corpus class, I declared a single attribute:

1. sentences: A list of Sentence objects containing all the sentences in the corpus.

Part 3:

For this part, I used the built-in library `glob` for getting the files' names into two lists of strings, one is for the XML file, and the other for the text files.

Then, I created the Corpus object and added all XML files and text files to it using the methods add_xml_file_to_corpus and add_text_file_to_corpus.

At the end, I created the output text file using create_text_file.

For the titles, I did not add them to the corpus since they have a lot of meaningless symbols. In addition, most of the times the title itself or at least most of its words are in the paragraphs following it. So, I think adding the titles to the corpus would not help, it would lead to noise in the corpus.

add_xml_file_to_corpus:

In this method, I used the bs4.BeautifulSoup to parse the xml file with the name received as an argument.

Firstly, I read the file and saved its data. Then, I parsed it and separated the data into heads and paragraphs. Then iterated over each sentence in the paragraph or head and split it to words. After that, I built the Token objects of each sentence, created the Sentence object based on the tokens in it and appended it to the sentences attribute in Corpus class.

add_text_file_to_corpus:

Firstly, I read the text file with the name received as an argument in utf-8. Then, saved the data as a list of lines which are not empty and are not headers.

After that, I iterated over the lines and split each of them to objects of Token, then created the Sentence object for each line and added it to the Corpus attribute called sentences.

create_text_file:

For this method, I iterated over the sentences in the corpus, then unpacked their tokens. Then, I added the tokens to the text I will be writing to the file separated by a single space. At the end of each sentence, I add a new line character to the text.

At the end, we wrote the data to the file with the name received as an argument in utf-8.