CL3.101 Computational Linguistics 1
Assignment 1

Course Instructor: Parameswari Krishnamurthy

Deadline: 31st January 2024, 11:55 PM

Instructions

- Your assignment must be implemented in Python.
- Do NOT use any standard library for tokenization. You are only allowed to use RegEx for this assignment.
- Make sure the submitted assignment is your original work. Do not copy any part of the assignment from your friends. Do not refer any AI systems to generate the code.
- No deadline extension will be possible. Please start early in order to finish it on time.
- Make sure to follow the submission format properly.

1 Tokenization: Using Regex

Tokenization is used in natural language processing to split paragraphs and sentences into smaller units that can be more easily assigned meaning.
For this assignment, you have been given a corpus [Refer to this for the corpus]. Build a Tokenizer using Regular Expression. You are expected to perform the following tasks on the provided corpus.

1.1 Task 1: Identifying Sentences

While analyzing a corpus, we first need to break it into sentences, i.e perform sentence-level tokenization. In this task, you have to perform sentence tokenization of the given corpus, and store the tokenized sentences following the format shown in the example. Use regular expressions to identify sentences in the provided text. For each identified sentence, create a separate `<Sent>` tag with a unique id. Display the text of the identified sentence under the corresponding `<Sent>` tag.

Example:

Input:
Hello, students! Welcome to the tokenizer assignment. This is your first assignment.

Output:

```
<Sent id="1">
    Text = Hello, students! Welcome to the tokenizer assignment.
</Sent>
<Sent id="2">
    Text = This is your first assignment.
</Sent>
```

1.2 Task 2: Identifying Words

After tokenizing the corpus at sentence level, we now proceed to word level tokenization. In this task, you have to perform word level tokenization on the sentences, i.e break the sentences into tokens, and store the tokens under the corresponding sentence id. Note that instances like ”they’re” should be split into ”they” and ”are”.

The output file for this task should look like:

```
<Sent id="1">
    Text = Hello, students! Welcome to the tokenizer assignment.
    Token 1 = Hello
    Token 2 = ,
    Token 3 = students
    Token 4 = !
    Token 5 = Welcome
    Token 6 = to
    Token 7 = the
    Token 8 = tokenizer
    Token 9 = assignment
    Token 10 = .
</Sent>
...
```

1.3 Task 3: Frequency Analysis

Now that you have tokenized the corpus at the word level, list the words along with their frequencies, in the format `<frequency> <token>`. You have to do this for the whole corpus, i.e give a list of frequencies of all unique words in the corpus.

The output file for this section should look like:

10 the
9 of
7 computer
...

1.4 Task 4: Type-Token Ratio

Calculate the type-token ratio for each sentence individually, as well as for the whole corpus provided to you. The output file for this section should look like:

```
<Sent id="1">
    Text = Hello, students! Welcome to the tokenizer assignment.
    Token 1 = Hello
    Token 2 = ,
    Token 3 = students
    Token 4 = !
    Token 5 = Welcome
    Token 6 = to
    Token 7 = the
    Token 8 = tokenizer
    Token 9 = assignment
    Token 10 = .
    Type-Token Ratio = 10/10 = 1.0
</Sent>
...
```

2 Submission Guidelines

Submit a single zip file on Moodle. Submit the codes and outputs for all the 4 tasks seperately, in four seperate files.

The name of the main file should be in the format [**RollNo_FirstName_RegexTokenizerAssignment.zip**]

Each folder corresponding to a task should contain the following:
Task_1:

- code.py or code.ipynb containing all the code.
- output.txt containing the tokenization output.

Also add a ReadME.md file along with the 4 files for the tasks in the main file, which will contain instructions to run the code and your assumptions (if any).