Habib University
CS- 458
Fall Semester 2024
Assignment 1

Q1-NLP Assignment: Resolving Ambiguities Between DD/MM/YYYY and MM/DD/YYYY Date Formats [40 points].

Objective:

The goal of this assignment is to practice resolving ambiguities between two similar date formats, DD/MM/YYYY and MM/DD/YYYY, using contextual clues or assumptions.

Task:

1. Write a Python script that reads the text from the provided `date_format_dd_mm_yyyy.txt` file. The file is available here.

2. Identify and extract all dates present in the text that follow the DD/MM/YYYY or MM/DD/YYYY format (you must use regular expressions for this task).

3. Attempt to determine whether each date is in DD/MM/YYYY or MM/DD/YYYY format based on:

- Contextual clues in the text (if available).
- Logical assumptions (e.g., there is no 13th month, so 13/05/2023 must be in DD/MM/YYYY format).
- You may also flag dates that are ambiguous and cannot be determined definitively.

4. Write the list of all identified dates along with your interpretation of their formats (e.g., DD/MM/YYYY or MM/DD/YYYY) to an output text file. Name the output file using your name and Habib's student ID (e.g., JohnDoe_Habib12345.txt).

Important Instructions:

You are not allowed to use any python libraries or modules that have a built-in implementation to identify the date formats. You can, however, use python libraries to read/write files or extract patterns (e.g., using regular expressions).

Deliverables:

1. Python script (.py file) with the implementation [10].
2. An output text file containing the extracted dates and their interpreted formats, named as per the instructions [10].
3. A report that includes [20]:

- Your proposed algorithm or method for resolving the date format ambiguity.
- A list of extracted dates and your interpretation of their formats (from the output text file).
- Any challenges or difficulties you encountered during the process, along with how you addressed them.

This instruction ensures students focus on developing their own methods to resolve the date format ambiguity without relying on built-in libraries for date parsing. Let me know if there are any other adjustments you'd like to make!

Q2-NLP Assignment: Identifying the First 10 Merges in a Wordpiece Algorithm [40 points].

Objective:
The goal of this assignment is to simulate the initial steps of a word piece algorithm by identifying the first 10 pairs of characters or subwords that would be merged.

Task:

1. Write a Python script that reads the text from the provided wordpiece_input.txt file. The file is available here.
2. Implement a simplified version of the word piece algorithm from scratch:
   - Start with individual characters as tokens.
   - Identify and merge pairs by prioritizes the merging of pairs where the individual parts are less frequent in the vocabulary. Use the following formula:
     - score=(freq_of_pair)/(freq_of_first_element×freq_of_second_element)
   - Continue this process until you have performed 10 merges.
3. Track and print the first 10 pairs that are merged by the algorithm.

Important Instruction:

- You must implement the wordpiece algorithm from scratch. Do not use any Python libraries or modules that have a built-in word piece implementation (e.g., Hugging Face, TensorFlow, etc.). The focus is on understanding and implementing the core logic of the algorithm manually.

Deliverables:

1. Python script (.py file) with the implementation [20].
2. A report that includes:

   - The first 10 pairs of tokens that were merged by your algorithm [10].

- Any challenges you encountered and how you addressed them [10].

This ensures that students gain hands-on experience with the wordpiece algorithm by implementing it from the ground up. Let me know if there's anything else you'd like to refine!

Q3-NLP Assignment: Tokenizing Urdu Text [20 points].

Objective:

The goal of this assignment is to practice reading and tokenizing Urdu text using Python.

Task:

1. Write a Python script that reads the text from the provided `urdu_text_input.txt` file. The file is available here.
2. Tokenize the text into individual words or tokens.
3. Write the list of tokens to an output text file (with each token on a separate line).

Important Instruction:

You are allowed to use any Python module or library for this task (e.g., `nltk`, `re`, `UrduHack`, etc.).

Deliverables:

1. Python script (.py file) with the implementation [5].
2. An output text file containing the list of tokens extracted from the Urdu text [5].
3. A report that includes:
    - Your python code [5].
    - Any challenges you encountered and how you addressed them [5].

This assignment will help students work with Urdu text data and explore tokenization techniques. Let me know if you need any further modifications!