Assignment 1: Correction Agent for ASR Errors in Voice-Enabled Assistants

Introduction

Voice-enabled assistants (e.g., Alexa, Google Home or other voice triggered applications such as search on a phone) rely on automatic speech recognition (ASR) to understand and respond to spoken commands by a person. ASR systems convert audio received from a human speaking to the device to the corresponding text for further processing. However, ASR systems can make mistakes that lead to errors in the output text extracted from the input audio. These errors can cause the assistant to misunderstand commands and potentially perform the wrong actions.

In this assignment we consider two main types of errors:

- **Similar-sounding characters** may be incorrectly recognised. For example, “The boy is eating” is recognized as *“The eoy is eating”* as the characters *'b'* and *'e'* can be mixed up.
- **Whole words** at the beginning or end of a sentence might be missing, where the speaker's voice might be softer. For example, *"I am going to the store”* is recognised as *“am going to the store”.*

In this assignment, your goal is to develop an agent that helps fix these errors by analysing the text and improving its accuracy. The agent will use a cost function that return *coherence score* (lower is better) for any text for a given audio. Your agent must use a search-based method that fixes the errors in the ASR output.

Problem Statement

The task is to develop a correction agent for text transcribed by an automatic speech recognition (ASR) system. The input to the agent is a text with all words in capital letters, separated by spaces. This text may contain errors due to the ASR

system, which can be incorrect character recognition or missing words at the start of end of the sentence as described before. To correct these errors, the agent will utilize two key resources:

- `phoneme_table.json`: This file contains information on which characters might be incorrectly recognised as others, providing possible substitutions that can occur.

```
{
  "K": ["G", "C"],
  "S": ["Z", "SH"],
  ...
  "KN": ["N"],
}
```

- `vocabulary.json`: This file lists the possible missing words that could appear at the start or end of the text received.

```
["RETURN'D", "DOCTORS", "INFANTRY", "DELIRIUM",..., "KILLED"]
```

Given an erroneous input text and the information about potential errors, there can be many possible corrections. To find the best correction, a search-based algorithm must be used. This algorithm should explore different correction options and use a cost function to evaluate them. The cost function determines how well each corrected sentence matches the original audio, with lower costs indicating more coherent and accurate sentences. The goal is to find the correction with the lowest cost, improving the accuracy of the ASR system's output.

Cost Model: In this assignment, the cost function is provided to you and is implemented using OpenAI Whisper model. Note that a detailed understanding of the Whisper model is not required for this assignment. Whisper is an ASR model that computes the likelihood of a text s for a given audio a. Specifically, it breaks down the text s into sequences of tokens text $[t_1, t_2, ..., t_n]$ where each token t_i consists of one or more characters. Then, Whisper computes the negative log likelihood as:

$$L(s,a) = -log(P_\theta(s|a)) = -\sum_{i=1}^{n} log(P_\theta(t_i|t_1, t_2, ..., t_{i-1}, a))$$

Here, θ denotes the parameters of the Whisper model which obtained by large-scale training. In essence, the Whisper model provides a cost function or a coherence score that relates the audio received with candidate text that your algorithm may consider. Formally, we the cost of a text s as for a given audio a is expressed as $f_{cost}(s) = L(s, a)$.

Implementation Details

- Implement a search-based algorithm to address the problem described above. Define the key components of the problem, including state representation, operators, transition costs, etc. Analyse the branching factor, or the number of possible corrections at each step.
- Uninformed search methods like DFS and BFS may struggle with longer sentences. Pathfinder algorithms like A* may also face challenges due to the lack of the goal state.
- Local search methods can handle larger problems, such as correcting sentences up to 100 characters within 120 seconds. However, careful consideration is needed to evaluate neighbouring states, and these methods may struggle with full exploration of the state space.
- You can start your implementation with local search methods and improve it further by performing graph-search style exploration of the search space.
- You must focus exclusively on search-based methods, as discussed in class, or their extensions and adaptations. Avoid using linear/integer programming or any domain-specific natural language processing techniques beyond search-based approaches. If you are unsure about any method, please consult the TA for clarification.
- Make sure your implementation **does not** make any assumptions about the cost function. During evaluation, a different cost function may be used.
- Submit your best algorithm as a single submission for this assignment.

Implementation Guidelines

- The starter code is provided for your implementation. The code is available on Moodle. The code has following structure.

```
A1
├── data
│   ├── phoneme_table.json  # Phoneme table with possible charac
│   ├── vocabulary.json  # List of possible words that can be mi
│   ├── data.pkl  # Sample data file containing audio, ASR incor
├── driver.py  # A driver code for running your implementation.
├── environment.yml # Conda environment file.
└── solution.py  # The file where you place your implementation
```

- You can run the driver program using following command. It will run your solution for each string in the data/data.pkl file and write the solutions to output.json.

```
python driver.py --output_file=output.json
```

- The file **solution.py** provides a template for your implementation. You are expected to implement the **asr_corrector()** method in the **class Agent**.
- solution.py is structured as below. asr_corrector() method is entry point for your implementation. It's your responsibility to store the best string you have got so far into **self.best_state** variable.

```
class Agent(object):
    def __init__(self, phoneme_table, vocabulary) -> None:
        """
        Your agent initialization goes here. You can also add co
        """
        self.phoneme_table = phoneme_table
        self.vocabulary = vocabulary
        self.best_state = None
```

```
    def asr_corrector(self, environment):
        """
        Your ASR corrector agent goes here. Environment object h
        - environment.init_state: Initial state of the environme
        - environment.compute_cost: A cost function that takes a

        Your agent must update environment.best_state with the c
        """
        self.best_state = environment.init_state
        cost = environment.compute_cost(environment.init_state)
```

- You must implement the following two functions in the Agent class.
  - `"__init__"`: Constructor of the class. The function inputs the phoneme table and vocabulary as dictionary of lists and list of string respectively. You can perform any preprocessing needed for setting up your agent.
  - `" asr_corrector "`: Your main implementation logic should go here. The function inputs an environment object which has following important members.
    - `init_state`: A string that holds the input text to be corrected.
    - `compute_cost`: A cost function that inputs any text and outputs the cost.
- Your implementation must store the final corrected text in `self.best_state` member of the Agent class.
- Ensure that your implementation runs efficiently and does not take an excessive amount of time to execute. During evaluations, our system may interrupt your code if it runs for too long. ***Therefore, we recommend periodically saving your best solution discovered so far in self.best_state.***
- **You must use Python 3.10 for your implementation. Your code should use only standard python libraries. Use of any other external libraries is not allowed.**
- The use of global variables or threading libraries is not permitted in your implementation.

Development Options

You can develop your solution either using your local setup or Kaggle. We recommend using Kaggle as it provide GPUs that can improve the cost function execution.

- Local setup: You can create a Python environment on your machine with `conda` using the command `conda env create -f environment.yml` . Here, `environment.yml` file is available with the starter code.
- Step-by-step process for evaluating on Kaggle is given in this video.

> https://prod-files-secure.s3.us-west-2.amazonaws.com/63d40f27-6659-4d11-b719-3313c51e3b49/dfee38a5-7ad0-4a23-8e4e-d83632bfe7ea/A1-Kaggle.mp4

Submission Instructions

- **This assignment is to be done individually or in pairs. The choice is entirely yours.**
- **The assignment is intended for both COL333 and COL671 students.** If the assignment is being done in pairs then the partner is to be selected only within your class. That is COL333 students should be paired within COL333 and similarly masters students in COL671 should pair within their class of COL671.
- Please submit only single algorithm (your best algorithm). Please succinctly describe the core ideas of your algorithm in the text file called **report.txt** (upto 2 pages in standard 11 point Arial font). The first line of the report should list the name and entry number of student(s) who submit the assignment.
- The assignment is to be submitted on **Gradescope**. Exactly ONE of the team members needs to make the submission. All registered students have already been added to Gradescope. The details are given below.
    - You need to upload a single ZIP file named `"submission.zip"`. Upon unzipping, this zip should create a folder containing `solution.py`, `report.txt` and `group.txt`. Do not rename the zip file.

- The group.txt should include the entry numbers of all the group members (one per line). If you are working alone, simply include your own entry numbers. A sample `group.txt` is given below.

  ```
  2020CSZ1234
  2020CSZ5678
  ```

- Gradescope will run a sanity check for your submission. Sanity check will test your code for incorrect Agent class specification and illegal imports and will run your code with dummy inputs. Your submission must pass this sanity test. Submissions failing the sanity check will not be evaluated.
- If you are working in pairs, you should select the partner using the "**Group Members**" option after uploading the submission in Gradescope.

- **The submission deadline is 6 PM on Thursday, September 5, 2024.**
- This assignment will carry $12(\pm 3)\%$ of the grade.

Other Guidelines

- Late submission deduction of (10% per day) will be awarded. Late submissions will be accepted till 2 days after the submission deadline. There are no buffer days. Hence, please submit by the submission date.
- Please follow the assignment guidelines. Failure to do so would cause exclusion from assessment and award of a reduction. Please strictly **adhere** to the input/output syntax specified in the assignment as the submissions are processed using scripts. Please do not modify files beyond the files you are supposed to modify. Failure to do so would cause exclusion from assessment/reduction.
- Queries (if any) should be raised on **Piazza**. Please keep track of Piazza posts. Any updates to the assignment statement will be discussed over Piazza. Please do not use email or message on Teams for assignment queries.
- **Please only submit work from your own efforts.** Do not look at or refer to code written by anyone else. You may discuss the problem, however the code implementation must be original. Discussion will not be grounds to justify

software plagiarism. Please do not copy existing assignment solutions from the internet or taken from past submission or submissions from other students; your submission will be compared against them using plagiarism detection software. Violations to the honour code will result in deductions as per course and institute policies.