HMM and Viterbi Algorithm Implementation

Overview

This project involves implementing a Hidden Markov Model (HMM) and the Viterbi algorithm. The primary objective is to determine the most likely sequence of hidden states given a sequence of observed outputs. Both first-order and second-order HMMs are implemented in this project.

Hidden Markov Model (HMM):

An HMM is a statistical model where the system being modeled is assumed to be a Markov process with hidden states. It consists of states, transitions between states, and outputs associated with the states.

File Structure

- DataSet.java: This class reads the input data file, parses it, and stores the information about states, outputs, training sequences, and testing sequences.
- Hmm.java: This class represents the first-order Hidden Markov Model, including methods for getting log probabilities of transitions and outputs.
- Viterbi.java: This class implements the Viterbi algorithm for finding the most likely sequence of hidden states.
- RunViterbi.java: This class contains the main method to run the Viterbi algorithm on the provided data set.
- Hmm2.java: This class represents the second-order Hidden Markov Model.
- Viterbi2.java: This class implements the Viterbi algorithm for the second-order HMM.
- RunViterbi2.java: This class contains the main method to run the Viterbi algorithm for the second-order HMM on the provided data set.

Usage

HMM and Viterbi Algorithm Implementation

1. Open the submitted ZIP file.

2. Open the source code in Visual Studio or any other IDE.

3. Open your terminal/console.

4. Check the files:

   - Type `ls` in the terminal.

   - You should see `1st-order` and `2nd-order` directories.

5. Compile the first-order files:

   - Navigate to the first-order directory by typing `cd 1st-order`.

   - Verify the contents by typing `ls` again. You should see some `.java` files and a `data` folder containing `.data` files.

6. Run the first-order files:

   - Type `make run-first-order` in the terminal.

   - The files will be compiled and executed automatically.

7. Change the data file for first-order:

   - Go to the `makefile` in the `1st-order` directory to specify the desired `.data` file as you want.

   - Run `make run-first-order` again to compile and run with the new data file.

8. Compile the second-order files:

   - Return to the main directory by typing `cd ..`.

   - Or you can just directly close and open the project again.

   - Navigate to the second-order directory by typing `cd 2nd-order`.

9. Run the second-order files:

   - Type `make run-second-order` in the terminal.

   - The files will be compiled and executed automatically.

10. Change the data file for second-order:

    - Go to `makefile` in the `2nd-order` directory to specify the desired file that you want (with .data

extension files).

- Run `make run-second-order` again to compile and run with the new data file.

Data Structures

- Map<String, Integer>: Used in DataSet.java to map state and output names to their respective indices.
- List<int[]>: Used to store sequences of states and outputs before converting them to arrays.

Notes

The provided `makefile` includes targets for compiling and running both first-order and second-order HMMs.