STDISCM Problem Set 4
Distributed AI System [100 pts]
By pairs

Background

Distributed systems enable tasks to be divided across multiple machines. In the case of AI applications, a client-server architecture can be implemented wherein the client-side software is responsible for collecting data, while processing and inference are performed by the server running on a more powerful computer.

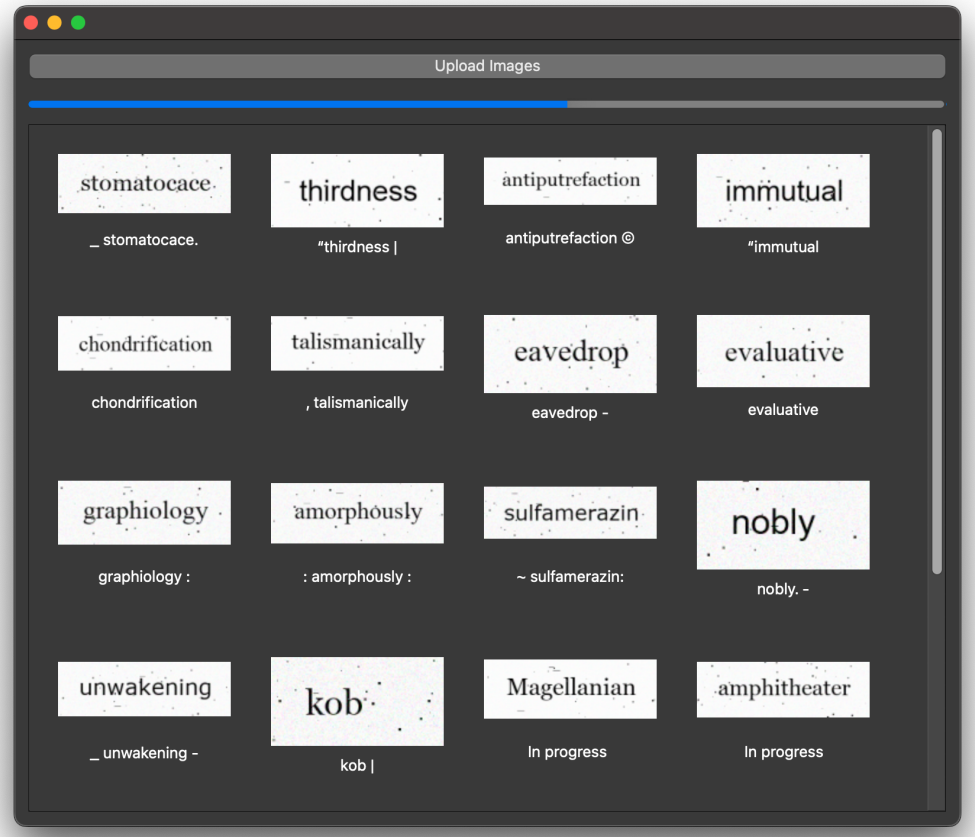
Objective

In this final problem set, you will be incorporating all the lessons you learned during the term. Your objective is to create a distributed OCR system with a client-server architecture:

- Client – This is a UI that allows users to upload images containing text for extraction. The uploaded images are sent to the server and the OCR results are sent back. Once the result for an image is returned, it should be shown right away even when the other results are pending. A progress bar should show how many of the uploaded images have been processed. As long as the progress bar has not reached 100%, users may continue uploading additional images which should also trigger the adjustment of the progress bar. Once all images have been completed (i.e. progress bar has reached 100%), subsequent uploaded images will be treated as a separate batch and previous results will be cleared.
- Server – This is a continuously running application that receives the images sent by the client, then passes them to one of its many threads running an OCR pipeline. Once the text of an image has been extracted, it should be returned right away to the client. The server application should run on a different machine or VM from where the client application is running.

You are not allowed to use any external library, other than the following: Qt Framework for the UI elements, Tesseract for the OCR, OpenCV for image processing (if you prefer it over Leptonica), and gRPC for interprocess communication.

Sample Application



Submission

Submit a Google Drive link containing the following project files:

- Presentation slides (in PDF format) that explains the following:
  - Multithreading
  - Synchronization
  - GUI Implementation
  - Interprocess Communication
  - Fault Tolerance (e.g. handling lost connections)
- A 5-10 minute MP4 video showing a demonstration of your system and explanation of the implementation details using your presentation slides as reference. Your demo should show various scenarios handled by your system. Only the demonstrated and explained features will be credited.
- Source code