Persado MLOps Take-home Exercise 2024 - Requirements

See also: Persado MLOps Take-home Exercise 2024 - Introduction

Exercise goal

This exercise's goal is to show the ability to use well known industry-standard technologies to deliver a solution to an application developer. Imagine that the application developer is building their own app which must depend on an ML system (which you will provide in this exercise) to provide suggestions to the end user. Specifically, the end–user wants to provide an English sentence that has a blank or missing part, and receive text suggestions that can fill in the blank in a way that makes the sentence sound positive. Examples:

Input: have a <blank> day
Output: good, excellent, amazing

Input: the application was <blank>
Output: well designed, so pretty, the best I've ever seen

Application specifications

The application should:

- Accept the input via an HTTP request
- Validate the input has the appropriate format
- Use a pre-existing NLP model pre-trained for fill-in-the-blank tasks to generate suggested phrases for completing the input (**NLP Task A**)
- Use a pre-existing NLP model pre-trained for sentiment analysis tasks to discard any suggestions that might sound negative. (**NLP Task B**)
- Return the suggestions in the HTTP response.

Suggestions

- Build a working application for NLP Task A first, then make the necessary changes for NLP Task B
- Use off the shelf components as much as possible, e.g.:
    - use the default model pipelines for each task from Huggingface Transformers,
    - use popular python frameworks such as Starlette or FastAPI for handling the requests and response,
    - Use battle tested web server implementations such as uvicorn.

Deliverables

Must-have

- A version controlled source code repository (e.g. on Github) with the application source code, in Python, that implements the exercise goal as a web server.
- Configuration for creating a container image which, when scheduled on Docker, Kubernetes, etc., launches the application for use in production.
- Basic documentation on how to build, run and use the application

Nice-to-have

- An automatic load testing/benchmarking solution (e.g. a Locustfile)
- A build system e.g. GitHub Actions, Jenkinsfile, etc. that builds the application container image.
- Deployment scripts and configurations that demonstrate how you would do load balancing of requests to increase scalability (e.g. with Kubernetes).
- Caching the request-response for increased response speed.
- Monitoring and Logging
- Authentication
- API Documentation