Embedding Upsampling

Overview

Hello! We are excited to share this take-home assignment with you. Please read the problem statement below and raise questions if any. We are happy to clarify over email or a quick call. Please try to get back to us in a week. After your solution is ready, please share it with us and schedule a 1-hour meeting to discuss your solution

Prompt

At Connectly, ML engineers focus on both modeling and infra work. This task is designed to test your ability to design, build, train, and ship ML models.

Task

Your task is to build a model and go through the full ML lifecycle. You're given a problem to solve with an ML model. You'll need to take the model through the full ML lifecycle from ideation to deployment.

ML Problem

We have taken 1536 Dimensional OpenAI embeddings and applied a mystery affine transformation to 32 dimensions. Your job is to build a model that given the 32-dimension embeddings can upsample them back to 1536-dimensions in a way that preserves the cosine angles between vectors of the original 1536-dimension embeddings. The upsampled vectors should be of unit length. This model is approximating the inverse of the mystery transform (mystery transform is not perfectly invertable).

We will provide**:**

- Training Embeddings: This contains 12130 32-dimensional unit length embeddings representing as a matrix of shape (12130,32).
    - File: https://cdn.connectly.ai/interview_prompts/ml_embedding_upsample/projected_train_embs.npy
- Cosine Angle Similarity Matrix: This is a (12130, 12130) where the value at position ij corresponds to the cosine angle <embedding_i, embedding_j>. Since these are unit vectors they were calculated as dot<embedding_i, embedding_j>. **These were calculated on the original 1536-Dimensional embeddings NOT the 32-Dimensional projected embeddings.**
    - File: https://cdn.connectly.ai/interview_prompts/ml_embedding_upsample/og_train_cos_theta.npy

Service

You should package the model as a service such that you can pass in 32 dimensional embeddings and it will return the upsampled embeddings (1536 dimensions). This service can be run locally - no need to deploy to a cloud provider (unless you want to).

The service has 3 request types:

1. Batch request: takes in a `.txt` file with 1 embedding per row and each value in the embedding is separated by a comma and writes the output upsampled embeddings to a `.txt` file. You must assume the file is too large to be read into memory.
    - Request args:
        i. input_filepath
        ii. output_filepath
    - Response args:
        i. batch_request_id: unique identifier that can be used to check the status of the batch request.
    - Example file 1 embedding per row with values separated by commas.

```
0.045, 0.34, 0.25, ..., -0.0094, 0.001
0.0035, -0.087, 0.1, ..., 0.011, 0.34
```

2. Batch status: returns the status of the batch job
    - Request args:
        i. batch_request_id
    - Response args:
        i. status: Literal[‘COMPLETED’, ‘IN_PROGRESS’, ‘FAILED’]
        ii. num_records_processed: the number of records processed
3. Realtime request: **optimize for low latency - the realtime request should be prioritized over long running batch requests**. If a long running batch request is in progress then it should be paused to prioritize the realtime request.
    - Request args:
        i. input_embedding
    - Response args:
        i. upsampled_embedding

We recommend starting with the simplest possible model and creating the service before optimizing the model.

Requirements

- You should complete this task within **4 hours.**
- Include instructions for running and interacting with the service.
- Include instructions for utilizing your training scripts.
- Upsampled embeddings should be unit vectors.
- You may use any available resources at your disposal. Including open source packages, chat-gpt, bard, stack overflow, etc…

- Please explain your design decisions and explicitly state the trade-offs you made.
- If you have time to optimize your model (optional): please include the graphs charts and data showing how you optimized your model.

Pair Programming Interview

When we meet to discuss the prompt, please make sure you can run your code. We will allocate 30 minutes to pair programming an extension to your service or model.