Data Science Coding Challenge

Write a simple HTTP API that returns embeddings for a given sentence.

Requirements

Using the universal sentence encoder hosted here create a python flask http api with 3 endpoints:

1. A `GET` endpoint at `/embeddings` that accepts a query parameter of `sentence` and returns a JSON encoded array of the embedding for the given sentence.

Example curl for requirement 1

```
$ curl "http://localhost:5000/embeddings?sentence=the+quick+brown+fox"

# response
{
  "embedding": [...]
}
```

2. A `POST` endpoint at `/embeddings/bulk` that accepts the JSON payload listed below and returns the embeddings for a list of sentences. The order of the embeddings returned should match the order that they are given in the request.

Example curl for requirement 2

```
# payload_2.json
{
  "sentences": [
    "the quick brown fox jumped over the lazy dog",
    "the five boxing wizards jump quickly"
  ]
}

$ curl -X POST -H "Content-Type: application/json" -d @payload_2.json
http://localhost:5000/embeddings/bulk

# response
{
  "embeddings": [
   [...],
   [...]
  ]
}
```

3. A `POST` endpoint at `/embedding/similarity` that accepts the JSON payload listed below and returns the cosine similarity between the embeddings of `sentence_1` and `sentence_2`

```
# payload_3.json
{
  "sentence_1":  "the quick brown fox jumped over the lazy dog",
  "sentence_2" : "the five boxing wizards jump quickly"
}

$ curl -X POST -H "Content-Type: application/json" -d @payload_3.json
http://localhost:5000/embeddings/similarity

# response
{
  "similarity": 0.28
}
```

Cosine similarity for vectors A, B

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|}$$

Documentation and boilerplate code for flask can be found here
Code should be sufficiently unit tested and appropriate abstractions used. Error conditions, such as invalid input JSON, should be responded to with relevant HTTP codes. Care should be taken to ensure that the tensorflow hub model is loaded and ready before the first request is served.

Any external libraries may be used but please ensure to include the libraries and their versions in a requirements.txt along with your submission. You can submit your solution as either a link to a public github repository or email us a zip archive.