HTX xData Technical Test Question – Data Scientist

Please follow closely to the Test Instruction

- You can make assumptions to any information not specified within the Test Instruction.
- You can make references to any other technical literature as required.
- Please complete the test and email the required deliverables within 7 calendar days upon receiving this email.
- Reach out to us if you have any queries via email.
- Submit Deliverables to the sender of this test.
- All submitted artifacts must be executable without errors. If a Python environment setup is required, please include the setup scripts in the submission. Failure to comply will result in the submission being considered incomplete.

The purpose of this test is to assess if you have the prerequisite technical ability. You have **a total of 7 calendar days** to complete the list of tasks below.

Complete all the following tasks (1 to 7) by yourself. You will need to document and comment your code properly and state any assumptions made within each task.

Tasks

1. Create a public git repository e.g., https://github.com/fred/myrepo. You may use any of the publicly available repositories like GitHub, Gitlab, etc.

    a) Add a `requirements.txt` for Python libraries
    b) Add a `.gitignore` file
    c) Add a `README.md` file to document the set up and run instructions for your code

2. Clone your repository from Task 1 and create a directory called `asr` in your repository. All code for this task should be kept in this directory. This task will require you to create a hosted microservice to deploy an Automatic Speech Recognition (ASR) AI model that can be used to transcribe any audio files.

    a) AI model to use: `wav2vec2-large-960h`
    https://huggingface.co/facebook/wav2vec2-large-960h

    This model is developed by Facebook and pretrained and fine-tuned on Librispeech dataset on 16kHz sampled speech audio. Please ensure that your speech input is also sampled at 16kHz. The reference link (above) includes the model card and its usage code.

    b) Write a ping API (i.e. http://localhost:8001/ping via GET) to return a response of “pong” to check if your service is working.

    c) Write an API with the following specifications as a hosted inference API for the model in Task 2a. Name your file `asr_api.py`.

url: http://localhost:8001/asr
Input parameter (content-type: `multipart/form-data`):

- `file` [string type] – the binary of an audio mp3 file

Response (content-type: `application/json`):

- `transcription` [string type] – the transcribed text returned by the model i.e., “*BEFORE HE HAD TIME TO ANSWER A MUCH ENCUMBERED VERA BURST INTO THE ROOM*”
- `duration` [string type] – the duration of the file in seconds i.e., “*20.7*”

Test command using CURL:

```
$ curl -F ‘file=@/home/fred/cv-valid-dev/sample-
  000000.mp3’ http://localhost:8001/asr
```

d) Data to use: `Common Voice`
Reference: https://www.kaggle.com/datasets/mozillaorg/common-voice

Download and use the following dataset using https://www.dropbox.com/scl/fi/i9yvfqpf7p8uye5o8k1sj/common_voice.zip?rlkey=lz3dtjuhekc3xw4jnoeoqy5yu&dl=0

Write a python file called `cv-decode.py` to call your API in Task 2b to transcribe the 4,076 common-voice mp3 files under cv-valid-dev folder.

Using `cv-valid-dev.csv`, write the generated transcribed text from your API into a new column called `generated_text`. Save this updated file in this folder.

e) Containerise `asr_api.py` using Docker. This will be in `Dockerfile` with the service name `asr-api`. Once the file is successfully processed, your code should delete the file.

3. Create a directory called `asr-train` in your repository. All code for this task should be kept in this directory. This task will require you to finetune an Automatic Speech Recognition (ASR) AI model.

a) From Task 2d, you are to use the common-voice mp3 files under cv-valid-train and `cv-valid-train.csv` for finetuning train dataset. Write a python jupyter notebook called `cv-train-2a.ipynb` for this task, using either TensorFlow or PyTorch. You are to split the dataset into 70-30 ratio where 30% is kept for training validation. You are to list down your explanation for your chosen preprocessing, tokenizer, feature extraction and pipeline processes (including hyperparameters selected). You are also required to visualise the training and validation metrics and explain your interpretation of these visualisations.

b) Rename your fine-tuned AI model: `wav2vec2-large-960h-cv`.

c) Within your jupyter notebook, `cv-train-2a.ipynb`, in task 2d, use your fine-tuned AI model to transcribe the common-voice mp3 files under cv-valid-test and compare the generated text against `cv-valid-test.csv`. Log your overall performance.

4. Compare your finetuned model from task 3 to be compared against the results from task 2a for the cv-valid-dev mp3 dataset. Describe your observations and propose a series of steps (including datasets and experiments) to improve the accuracy. Your answer can be saved as `training-report.pdf` under the main repository.

5. Create a directory called `hotword-detection` in your repository. All code for this task should be kept in this directory. This task will require you to detect hot words.

a) Using the transcribed results from cv-valid-dev mp3 dataset using your finetuned model in task 4, the hot words to be detected are: "`be careful`", "`destroy`" and "`stranger`". Save the list of mp3 filenames with the hot words detected into `detected.txt`. Write a python jupyter notebook called `cv-hotword-5a.ipynb` for this task.

b) Text embedding model to use: `hkunlp/instructor-large`
https://huggingface.co/hkunlp/instructor-large

Using the text embedding model, write a python jupyter notebook called `cv-hotword-similarity-5b.ipynb` to find similar phrases to the 3 hot words in task 5a. Using `cv-valid-dev.csv`, write the Boolean (true for a record containing similar phrases to the hot words; false for a record that is not similar) into a new column called `similarity`. Save this updated file in this folder.

6. Essay question – please read https://arxiv.org/pdf/2205.08598.pdf and propose a model self-supervised learning pipeline to cater dysarthric speech and describe how you would do continuous learning in 500 words. Your answer can be saved as `essay-ssl.pdf` under the main repository.

7. Once you have completed this test, submit your git repository url via email. You may remove your repository once you have received confirmation that your test submission has been received and reviewed.