Data Science Project (2024) - Sentiment Classification API template

This repository contains boilerplate code for creating your own sentiment classification restful API and deploying it to a public cloud service provider. After following the instructions below, you should have a public URL for your API that can be used to call your ML model and to classify the sentiment in any text documents your provide it with.

Note: This guide assumes that you have already developed your ML model and packaged it into a `.model` file according to the instructions provided in the course.

🛠️ Development instructions

Note: The following instructions assume that you are working on one of the VU compute hub servers within the `Data Science Project` environment.

- Copy your `.model` file into the main directory of this folder.
- Open the `app.py` file and adjust the variables `GROUP_ID`, `MODEL_FILE` and `MODEL_VERSION` to reflect your GroupID, the name of your model file (including .model extension) and the version number of your model (you can choose this as you like to keep track of different versions of the model you have developed).
- Adjust the code inside the function `batch_predict(model, items)` so that it uses the model in your model file to make a prediction for each supplied document. Make sure that you do not change the return value of the function, it should still return a list of dictionaries with the `id` keys of the documents and and the corresponding predicted integer `label`.
- Open a new terminal and switch to a Bash shell by typing `bash`. Start your API development server by running

```
python app.py
```

Note that you can exit the server using `Ctrl + C`.

- While your development server is running, click the `+` tab and open a new Terminal and enter `bash`.
- In the second terminal, run the following command to query your API using a `GET` request using the `curl` command.

```
curl localhost:8000
```

You should see that it returns a JSON response with the metadata of your model, similar to the output below.

```
{
  "meta": {
    "groupID": "caffeinated-chameleons",
    "modelFile": "benchmark.model",
    "modelVersion": "v0.2",
    "pythonVersion": "3.11.8"
  }
}
```

Verify that the output is correct.

Next, we are going to test if our API can successfully classify text documents that we send to it. Run the following command to use the `curl` utility to send a `POST` request to our API, asking it to classify a document with the text `I love my new sentiment classification API!`.

```
curl -d '{"items": [{"id": "test", "text": "I love my new sentiment classification API!"}]}' -H
"Content-Type: application/json" -X POST localhost:8000 | jq
```

If you implemented your model and API correctly, you should see your API return a JSON response with the predicted sentiment label for the document.

```
{
  "items": [
    {
      "id": "test",
      "label": 1
    }
  ]
}
```

If the command throws an error, you can switch to the other terminal which will display the error message that you can use to debug your code.

You can now exit both of the Terminals.

- Once your API is working, we will finally verify that we have all the files in place to deploy our model in the next step. Open a new Terminal, navigate to the folder where your API code is stored. Switch to `bash` and run the following command

```
gunicorn app:app
```

You should see output similar to the following

```
[2024-06-01 07:55:30 +0000] [832] [INFO] Starting gunicorn 22.0.0
[2024-06-01 07:55:30 +0000] [832] [INFO] Listening at: http://127.0.0.1:8000 (832)
[2024-06-01 07:55:30 +0000] [832] [INFO] Using worker: sync
[2024-06-01 07:55:30 +0000] [833] [INFO] Booting worker with pid: 833
```

You can now repeat the steps above in a new Terminal to query your API using `GET` and `POST` requests with the `curl` utility. Once you have verified that everything is working, you can exit gunicorn with `Ctrl + C` and close both Terminals.

- 🎉 Congratulations! You have successfully implemented your API and are now ready to deploy it to a public cloud service provider by following the instructions below.

🚀 Deployment instructions

🚨 Make sure you complete all steps of the Development instructions above before following the deployment instructions. You should have successfully tested your API locally before you deploy your API.

Note: For simplicity, the instructions below are written under the assumption that you are deploying your API to the `Koyeb` (https://koyeb.com) public cloud service provider. However, you are free to choose any cloud service provider you would like and appropriately modify the instructions. Once advantage of Koyeb is that you do not need to provide any payment information when creating a Starter/Hobby account and can use their low power `free` instances (which should be enough for the purposes of this course project).

- Sign up for a `Starter` account at https://koyeb.com using the `Hobby` plan (no credit card required).
- Choose a name for your organization (e.g. your groupID). The name will be part of your API URL so make sure it does not contain any personal identifying information (PII).
- Create a Personal Access Token in your Account Settings at https://app.koyeb.com/user/settings/api. Carefully save the token, it will only be displayed once. If you loose your token, you need to revoke it and generate a new one.
- On one of the compute hub servers in the `Data Science Project` environment, open a new Terminal and run the following command to install the `Koyeb CLI`. This will install the CLI in the `bin` folder in your home folder on the servers.

```
export KOYEB_INSTALL=$HOME
curl -fsSL https://raw.githubusercontent.com/koyeb/koyeb-cli/master/install.sh | sh
```

- Enter a bash console by typing `bash`. Log into your Koyeb account by running `koyeb login`. When prompted, provide the email address and the Personal Access Token you generated previously. Once you are logged in successfully, your credentials will be stored in `~/.koyeb.yaml`, so you do not need to log in every time you use the Koyeb CLI.
- Navigate to the folder where your API code is stored using the `cd` command. Run `ls` and make sure you see at least the following files.

```
<yourmodelfile>.model  Procfile  app.py  requirements.txt  runtime.txt
```

- Run the following command to deploy your API to a new Koyeb app called `dsp-2024`, creating a service named `nlp-api`.

```
koyeb deploy . dsp-2024/nlp-api --instance-type free --regions fra
```

- Go to the Koyeb control panel in your browser. If everything was successful, you should see the new app and service showing up. The deployment will be in a `Provisioning` state and it will take around 5min for your API to be fully deployed. Once the app is deployed and started, you can copy the public URL of your app (something similar to https://--.koyeb.app/).
- If you open this link in your browser, you should see the meta-data of your model being returned.
- You can also test your API by running a POST request against your service using the `curl` command, asking it to classify an example sentence.

```
curl -d '{"items": [{"id": "test", "text": "I love my new sentiment classification API!"}]}' -H
"Content-Type: application/json" -X POST https://<org-name>-<app-name>-<id>.koyeb.app/ | jq
```

- You should see a JSON response from your API with the predicted label

```
{
  "items": [
    {
      "id": "test",
      "label": 1
    }
  ]
}
```

- 🎉 Congratulations! You can now submit your API to the leaderboard using your API URL and the PIN you can find on Canvas.