CS 5293, Spring 2022 Project 2

Cuisine Predictor

In this project, imagine you are a data scientist for a hotel chain. This hotel chain is well-known for the variety and flavor of its food. Your marketing department has reported that tweaking the menu can improve success by 15%. As the chief data scientist, you are given the task of improving menu profits. To do this, you must help the hotel staff with menu planning and preparation. The goal of this project is to assist the Executive Chef in understanding the large menu set better by providing a cuisine predictor.

You are given the master list of all possible dishes, their ingredients, an identifier, and the cuisine for thousands of different dishes. You realize that if you cluster foods by their ingredients you can help the restaurant change foods but keep the ingredients constant. You present a display of clustered ingredients and train a classifier to predict the cuisine type of a new food. To complete this project, you will be asked to create a small interface that asks a user to supply a list of ingredients and returns the predicted cuisine type.

The data sets for this project is provided by Yummly.com — it was used as part of a Kaggle contest (some people in the course have already participated in this contest.) Below are the requirements for this project. To help the development of your code, test on a subset of the data first. Make sure to start as soon as possible!! You are free to discuss approaches using the help board, but solutions must be your own. I also encourage reaching out and asking questions early as many students will have similar concerns.

Here are two prominent papers that analyzed food data. We used data from this document in previous classes.

1. Flavor Network and the Principles of Food Pairing
2. Inferring Cuisine - Drug Interactions Using the Linked Data Approach

Project 2 Specification

The goal of project 2 is to (1) create applications that take a list of ingredients from a user and attempts to predict the type of cuisine and similar meals. (2) Consider a chef who has a list of ingredients and would like to change the current meal without changing the ingredients. The steps to develop the project should proceed as follows.

1. Train (or index for search) the food data in the provided.
2. Ask the user to input all the ingredients that they are interested in (through command line arguments).
3. Use the model (or search index) to predict the type of cuisine and tell the user.
4. Find the top-N closest foods (N defined with a command line parameter). Return the IDs of those dishes to the user. If a dataset does not have IDs associated with them you may add them arbitrarily.

Click here to access the yummly.json data set. The yummly data set contains sets of foods and a cuisine. You can use the cuisine type as a label.


```
{
  "id": 10276,
  "cuisine": "mexican",
  "ingredients": [
    "chili powder",
    "crushed red pepper flakes",
    "garlic powder",
    "sea salt",
    "ground cumin",
    "onion powder",
    "dried oregano",
    "ground black pepper",
    "paprika"
  ]
}
```


There many ways to complete the project. I encourage you to be creative and take advantage of the tools you have learned to design your system.

Below is the expection methood of execution.

Note that `project2.py` is in the main folder. Multiple `--ingredient` flags can be added to specify the ingredients that should be added to the meal. If an ingredient has multiple words use quotes as in the “rice krispies” example.

```
pipenv run python project2.py --N 5 --ingredient paprika
                                    --ingredient banana
                                    --ingredient "rice krispies"
```

In this example, 3 ingredients (paprika, banana, and rice krispies) were passed in to the program. The program then rand and matched the closes cuisine type, followed by a distance to this cuisine. The next line has the list of the top 5 closest meals with their distance in parenthesis. This is a synthetic example, feel free to use creative solutions. With an example output as follows.

Your project should return the same json formatting. The closest cuisine is listed as the value of the key “cuisine”. The similarity score of the closest cuisine is a double value for the key “score”. The key “closest” has a value that is an array of dictionaries with an “id” for the cuisine id and the similarity score “score”. The array should have N entries, given but the commandline paramter `--N`.

This output shows (a) the closest cuising and second the top-N cuisine dishes.


```
{
  "cuisine": "America",
  "score": 0.91,
```

```
  "closest": [
    {
      "id": "10232",
      "score": 0.34
    },
    {
      "id": "10422",
      "score": 0.15
    },
    {
      "id": "45",
      "score": 0.13
    },
    {
      "id": "7372",
      "score": 0.04
    },
    {
      "id": "9898",
      "score": 0.02
    }
  ]
}
```

Create a README that allows us to understand all assumptions and design decision you encountered when developing the system. It should also include directions on how to recreate and run your applications. Use the python project structure that we used in previous assignments. Include tests to validate your code. You will submit the README along with all source files to gradescope. You should also give the professors access to your GitHub repository.

Create a repository for your project on GitHub

Create a private repository GitHub called `cs5293sp22-project2`. **Note that is the repository is public you may be considered academic dishonesty.**

Add collaborators `cegme`, `jasdehart` by going to `Settings > Collaborators`.

We will be testing your code on the VM instances described in class. Please ensure your code runs correctly on the instances. If any extra information is necessary such as an extra large instance size you must specify that in your instances.

You should regularly `git add <file>`, `git commit -m`, and `git push origin main` your code changes to GitHub.

Submitting your code

When ready to submit, create a tag on your repository using git tag on the latest commit:

```
git tag v1.0
git push origin v1.0
```

Version v1.0 lets us know when and what version of code you would like us to grade. If you need to submit an updated version, you can use the tag `v1.1`. If you would like to submit a second version, use the tag `v2.0`.

If you need to update a tag, view the commands in the following StackOverflow post.

Rubric

Criteria	Exemplary	Exceeds Expectation	Meets Expectation	Developing	Unacceptable
Cluster Documents	Usage and implementation of clustering algorithms is complete and makes note of each package and implementing them to convert text into features. Implementation of feature transformation is rich. The README is complete, justifies all assumptions and allows reproducibility.	Implementation of clustering algorithms is complete. Generation of text into features is complete. The README is complete ands allows reproducibility.	Implementation of clustering algorithms is partially complete. Generation of text into features is partially complete. The README is complete, describing functions.	Usage and implementation of any clustering algorithm is done partially but makes note of wrong packages for converting text into features. Implementation of feature transformation is not done perfectly. The README is partially developed.	Usage and implementation of any clustering algorithms contains significant errors. Text-to-feature transformation is not done. The README is not completed.

Criteria	Exemplary	Exceeds Expectation	Meets Expectation	Developing	Unacceptable
Creating an Interface	Interface takes ingredients from a user and uses a model to predict the type of cuisine and tell this to user to find top N closest foods. Return IDs of these dishes to user. All errors by the user are handled. All methods are efficient, and the processes is effectually presented to the user.	The user can easily enter ingredients. The method used to predict the type of cuisine is sound. The method to find top N closest foods is quick. Return IDs are returned to the user. Some user errors are handled.	Interface created to take ingredients from a user. A coherent model is created to predict the type of cuisine and tell this to user to find top N closest foods. IDs of these dishes are correctly returned to user.	Interface cannot correctly retrieve information or needs better input from the user. The interface may look good but predicting the type of cuisine has significant errors in it. Discovery of the top N closest foods is incomplete.	Interface is confusing or not completely done. Interface will have lot of errors in predicting the type of cuisine and the finding the top N closest foods is also not implemented.

Helpful links

Below is a link to the yummly data set and other files that may be helpful as additional data that you *could* use to train the models.

- yummly.json
- srep00196-s2.csv
- srep00196-s3.csv

Here are local copies of the research papers listed earlier.

- Fiavor Network (pdf)
- Inferring Cuisine (pdf)

Submission Instructons

When submitting your project please follow the intructions closley and make the grading of the project easier. We will be running your code on the linux system described in class. Test your code on this system.

README.md

The README file name should be *uppercase* with an `.md` extension. You should write your name in it, an example of how to run it, and a list of any web or external resources that you used for help. The README file should also contain a list of any bugs or assumptions made while writing the program.

Note that you should not be copying code from any website not provided by the instructor. You should include directions on how to install and use the code. You should describe any known bugs and cite any sources or people you used for help. Make sure you discuss your test functions. **Be sure to include any assumptions you make for your solution.**

COLLABORATORS file

This file should contain a comma-separated list describing who you worked with and a small text description describing the nature of the collaboration. This information should be listed in three fields as in the example is below:

```
Katherine Johnson, kj@nasa.gov, Helped me understand calculations
Dorothy Vaughan, doro@dod.gov, Helped me with multiplexed time management
```

Project Descriptions

Your code structure should be in a directory with something similar to the following format:

```
cs5293sp22-project2/
├── COLLABORATORS
├── LICENSE
├── README
├── Pipfile
├── docs/
├── project2.py
├── setup.cfg
├── setup.py
└── tests/
    └── ...
```

setup.py

```
from setuptools import setup, find_packages

setup(
        name='project2',
        version='1.0',
        author='You Name',
        authour_email='your ou email',
        packages=find_packages(exclude=('tests', 'docs')),
        setup_requires=['pytest-runner'],
        tests_require=['pytest']
)
```

Note, the `setup.cfg` file should have at least the following text inside:

```
[aliases]
test=pytest

[tool:pytest]
norecursedirs = .*, CVS, _darcs, {arch}, *.egg, venv
```

Tests

The general rule is you should aim to have a test for each feature. Including tests help people understand how your code works, in addition to verifying assumptions during development. Tests should be runnable by using `pipenv run python -m pytest`. If your tests cannot be run with the command above, you may lose points.

Do not hard code any paths or directories.
If you are using an external module, make sure you code installs the module. You can make your Pipfile install spaCy models by adding a line to your Pipfile under packages. An example is below:

```
[packages]

spacy = "*"
en_core_web_md = {file = "https://github.com/explosion/spacy-models/releases/d
```

In the code you could use the model as follows:

```
import spacy
import en_core_web_md
# ...

  nlp = en_core_web_md.load()
  # ...
```

You may have to `pipenv update` after adding the model.

Grading

Grades will be assessed according to the following distribution:

- 60%: Correctness.
  - This will be assessed by giving your code a range of inputs and checking the output.
  - Use the creation of tests to prove correctness.

- 40%: Documentation.
  - Your README file should fully explain your process for developing your code.
  - All other commands should be well-documented.

Note we will be running your code in batch it is important that you follow directions closely

- 2022-04-08 Saving/Persisting sklearn models

Back to Project List
