Educational Web Documentation

1) An overview of the function of the code (i.e., what it does and what it can be used for).

EducationalWeb (EW) was already rich in features. Two big features were the slide search, and term definitions, which were powered by ElasticSearch (ES) and MetaPy respectively. Our goal was to add a Piazza integration into the app. For this we re-used some of the existing components and added a Piazza dataset to them.

First, we created a Python scraper that downloads all the course posts from Piazza and organizes them. This scraper can be found under "/piazza/piazza_downloader.py". At runtime, this code creates the "/piazza/downloads" folder, which contains a set of JSON files. Next, we extended the "create_es_index.py" code to create a new "piazza" index and ingest the dataset into ES. It also creates a MetaPy compatible dataset under "/piazza/index".

In the app, we repurposed some of the already existing code (which is used to search for key terms in slides) to now also search for relevant Piazza posts that have been ingested into ES. The results are displayed at the bottom of the page and link to the correct Piazza posts.

Currently, each page also displays relevant slides on the right sidebar of the page. Since we wanted to explore Okapi-BM25 and its performance on the Piazza dataset, we made use of this area and are now displaying relevant Piazza posts, based on the content of the current slide. Initially, we wanted to use MetaPy for this, as it was already used by on component in the app. However, MetaPy does not work with Py3.x (Py2.x has been retired), and search queries will hang indefinitely. This is why we opted to replace MetaPy with a Python library called "rank-bm25"; A high performance Py3.x compatible implementation of Okapi-BM25. After swapping out MetaPy for the new library in the existing code, we added a new API POST endpoint at "/rank_piazza". After calling this endpoint with search terms, the API will return the top 20 relevant Piazza posts. On the front-end we then added an AJAX request to this endpoint and pass along the content of the current slide. This allows us to find relevant Piazza posts for the current context and help the user stay on track while they are, for example, studying.

2) Documentation of how the software is implemented with sufficient detail so that others can have a basic understanding of your code for future extension or any further improvement.

The app requires Py3.7 to run, and according package requirements have been added to "requirements.txt". With exception of MetaPy, the already existing code did not change, so past documentation still applies. These are the most significant changes:

1. New folder called "/piazza" containing all the code used for Piazza dataset management
2. Additional index and data for ES by extending "create_es_index.py"
3. Additional search functionality on the front-end to retrieve Piazza posts by search terms
4. MetaPy to "rank-bm25" migration
5. New endpoint "/rank_piazza" to retrieve relevant Piazza posts based on current slide content

Opportunities for further extension:

1. Make better use of Piazza references -- The piazza parser that was created generates a list of references for the current post. Some of these references are to other piazza posts and other of the references are to images or files attached to the post. This presents two opportunities for extension: 1) To display the referenced posts in the search results and 2) to also download the referenced files and add their text to the EducationalWeb indices. See get_references in the Entry class --

https://github.com/pgruber2/EducationalWeb/blob/36e90933330ef606a54083feb78b9966035ee41a/piazza/piazza_post_parser.py#L98

2. As part of our project, we conducted a survey focused on Piazza and EW. We used the information collected to confirm that the search results returned by our project would align with students' expectations. Specifically, one of the key results was that student's value instructor posts and posts with endorsements. This is why we draw attention to posts with these attributes when displaying the results. The survey has a lot more information which could be analyzed to improve EducationalWeb. For an overview of the survey see -- https://uillinoisedu-my.sharepoint.com/:v:/g/personal/jjhenn2_illinois_edu/Ean9a3GjBiFIq1PY5edZ8usBhEnp-aUwb4oOH3KXfxEheA?e=iXvtQ4

3) Documentation of the usage of the software including either documentation of usages of APIs or detailed instructions on how to install and run a software, whichever is applicable.

Please see https://github.com/jessehenn/CourseProject/blob/main/README.md for the overview of all key deliverables.

For the ease of verifying all requirement have been fulfilled these instructions specific to setting up the project have been duplicated from https://github.com/pgruber2/EducationalWeb/blob/master/README.md

EducationalWeb

This project currently required Python 3.7.9 and the instructions below have successfully been completed on MacOS. NOTE: The 3.7.9 requirement is currently a strict requirement because of portions being dependent on Metapy.

Starting Structure of the Project

These instructions assume you have created a **project** directory and have the EducationalWeb directory as a sub-directory.

```
cd project
tree -d -L 4
```

```
.
└── EducationalWeb
    ├── log
    ├── para_idx_data
    ├── paras
    │   └── inv
    ├── paras_nohead
    │   └── inv
    ├── pdf.js
    │   ├── build
    │   ├── docs
    │   ├── examples
    │   ├── extensions
```

```
│   ├── external
│   ├── l10n
│   ├── node_modules
│   ├── src
│   ├── static
│   ├── test
│   └── web
├── piazza
├── slides
├── static
└── templates
```

Download EducationalWeb Dependencies

1. Download tfidf_outputs.zip from here -
   - https://drive.google.com/file/d/19ia7CqaHnW3KKxASbnfs2clqRIgdTFiw/view?usp=sharing
2. Download cs410.zip from here -
   - https://drive.google.com/file/d/1Xiw9oSavOOeJsy_SIilxPf4aqsuyuuh6/view?usp=sharing
3. Download lemur-stopwords.txt from here -- https://raw.githubusercontent.com/meta-toolkit/meta/master/data/lemur-stopwords.txt

Place EducationalWeb Dependencies

1. `tfidf_outputs.zip` needs to be unzipped and placed under `EducationalWeb/static`
2. `cs410.zip` needs to be unzipped and place under `EducationalWeb/pdf.js/static/slides`
3. `lemur-stopwords.txt` needs to be placed under `EducationalWeb`

When this is done your structure will look as follows:

```
.
├── project
│   ├── EducationalWeb
│   │   ├── lemur-stopwords.txt
│   │   ├── pdf.js
│   │   │   ├── static
│   │   │   │   └── slides
│   │   │   │       └── cs-410
│   │   │   │           ├── 01_orientation----01_orientation-information----
01_course-introduction-video_410DSO-intro.txt
│   │   │   │           │   ├── cs-410----01_orientation----01_orientation-
information----01_course-introduction-video_410DSO-intro.txt----slide0.pdf
│   │   │   │           │   ├── ...
│   │   │   │           ├── ...
│   │   ├── static
```

```
│   │   │   └── tf_idf_outputs
│   │   │       ├── idf_list.p
│   │   │       ├── normalized_tfidfs.npy
│   │   │       ├── normalized_title_tfidfs.npy
│   │   │       ├── ss_corpus.p
│   │   │       ├── vec.p
│   │   │       └── vocabulary_list.p
```

Install gulp

MacOs Instructions

```
brew install gulp-cli
```

I'm sure there is a way to install it from npm as well (feel free to add instructions here).

Run the Gulp server

From `EducationalWeb/pdf.js/build/generic/web` , run the following command: `
`gulp server`

Create a 3.7.9 Python Virtual Environment

Assuming you have Python 3.7.9 installed and accessible using **python3** from the command line

At a second command line:

```
cd project
python3 -m venv venv-3.7.9
source ./venv-3.7.9/bin/activate
cd EducationalWeb
pip install -r requirements.txt
```

The rest of these instructions assume you are using the python virtual environment created in this step.

Downloading Piazza Posts

We need to download the piazza posts before we can use them with EducationalWeb. For additional details you can refer to piazza resources. In the command line below use your email and piazza class id.

```
cd piazza

export SLEEP_OVERRIDE=1
```

```
./piazza_downloader.py jjhenn2@illinois.edu kdp8arjgvyj67l
```

When this step is complete the piazza directory structure is updated to look like this:

```
│   ├── EducationalWeb
│   │   ├── piazza
│   │   │   ├── downloads
│   │   │   │   └── kdp8arjgvyj67l
│   │   │   │       └── 2020_12_05
│   │   │   │           ├── post_1000_kh0o8sghyod2dr.json
│   │   │   │           ├── post_1001_kh0ol07fn2436l.json
│   │   │   │           ├── post_1002_kh0rp6a3nja5ue.json
│   │   │   │           ├── ...
```

Install and run ElasticSearch

- This project is using docker for ElasticSearch. You can use https://docs.docker.com/get-docker/ to install docker. These instructions have been tested with Docker 2.5.0.0 on MacOs.

Use the following command to start ElasticSearch at a third the command line.

```
make up
```

Create the index in ElasticSearch by running

At the second command line create the ElasticSearch index using the virtual environment created earlier.

```
source ./venv-3.7.9/bin/activate
cd EducationalWeb
python create_es_index.py
```

NOTE: If you need to regenerate the index due to newly downloaded piazza posts you can use `curl -XDELETE localhost:9200/piazza`

Start the flask server

With the second command line used to create the ElasticSearch index start the flask sever.

```
python app.py
```

Congratulations

Once you made it this far you are ready to use EducationalWeb.

The site should be available at http://localhost:8096/

4) Brief description of contribution of each team member in case of a multi-person team.

The work was split evenly between all the team members. We'll mention the team member and reference the changes from section 2 next to the name.

- Jesse Henn – jjhenn2 (Team Captain): 1, 3 + various work on documentation, managing reports, etc.
- Patrick Gruber – pgruber2: 2, 4, 5