CS410 Project Documentation

“TIS Chicago”

https://github.com/JosephineFalso/CourseProject

Josephine Falso

Sree Alaparthi

Narasimha Kethireddy

Project Overview

Our group has built a back-end application to cohesively integrate the instructor's Coursera lecture videos with additional information about significant terms in the lectures. The CampusWire postings and the course textbook were mined for references to these significant keywords.

Our application provides a text area on the webpage to display the most significant keywords from the course material for each week. Following the keyword, the user will find links to CampusWire postings that reference the keyword as well as the page numbers from the course textbook for this keyword and a link to the course text.

While the course videos and transcripts for the entire course are available on the website, the keyword search functionality has been completed for the content in Week 3. Our code could be enhanced to include keyword functionality for additional weeks of course material in a future release.

Our application alleviates the inconvenience of navigating in and out of Coursera to resolve questions about new concepts and terminology. By incorporating all of the relevant information directly on the webpage, the user will have a more focused and productive learning experience. The techniques applied to this project demonstrate mastery of both text access and text mining.

Project Challenges

The team could not scrape the video transcription text from Coursera directly because it is behind the login screen. Recaptcha is used for Coursera login, and the Coursera Terms of Service prohibit bypassing captchas. Therefore, the TA staff provided the scraped text for our use.

CampusWire does not have public APIs exposed to fetch the data. Additionally, there is not an option provided in the UI to export the CampusWire data dumps. Therefore, the developer had to obtain the data manually by copying the responses from the Developer Tools Network Traces.

Software Implementation

Week3_Coursera_Text.txt

The scraped transcription text from the Coursera Week 3 lecture videos is located in /Scraped_Text/Week3_Coursera_Text.txt. This is the input for the generate_topic_term.py script. The scraped text was provided by the TA staff, and the text for the lecture videos from week 3 was extracted from the larger file into Week3_Coursera_Text.txt.

generate_topic_term.py

The scraped transcription text from the lecture videos is processed via the script /Generate Keywords/generate_topic_term.py. The python library NLTK is used to obtain a list of the most commonly occurring unigram terms in the week 3 lecture videos. Upon evaluation of the most common terms by a human user (/Text_Output/CommonWords_Week3.txt), the team determined that some important topics were not explicitly mentioned frequently enough to be returned in a list of the most common terms. Therefore, a "gold standard" list of the eight key topics from week 3 of the lecture videos was placed into /Text_Output/TopicList_Week3.txt.

campuswiresearch.py

The eight key topics from the week 3 lecture videos are mined from CampusWire postings via the script /src/campuswiresearch.py. The input is the json file for each term from the /campuswire_raw_data/ folder, which represents the raw data extracted from CampusWire. The /campuswire_raw_data/ folder is populated by the CampusWire Search API.

The script parses the json structures and extracts the topic, URL of the posting, and ranked relevance of the posting. The json output file for each term is placed into /output.

The python script is exposed as a lambda service on AWS. Further information on deploying the service to AWS is provided in the Software Usage section below. It is invoked from the user interface.

pdfsearch.py

The eight key topics from the week 3 lecture videos are mined from the course textbook via the script /src/pdfsearch.py. The python library PyPDF2 is used to mine the pdf file. Each page of the course text is searched in an iterative manner to locate the eight key topics from the week 3 lecture videos. When a keyword is located, it's page number is appended to a list of page numbers for each term. The page numbers and link to the course text are returned in a json format.

The python script is exposed as a lambda service on AWS. Further information on deploying the service to AWS is provided in the Software Usage section below. It is invoked from the user interface.

API Specifications

There are 3 APIs developed for this project.

i) searchCampusWire - Returns the title, body summary text, URL of the Campuswire posts for a given keyword. The list of keywords for which the data is downloaded from Campuswire are precision, recall, average precision, f-measure, map, gmap, dcg, ndcg and mean average precision.
   - Endpoint: https://7dbuuwphha.execute-api.us-east-2.amazonaws.com/myDeployment/searchCampusWire?term={keyword}
   - Replace keyword with the actual keyword such as precision. For example, https://7dbuuwphha.execute-api.us-east-2.amazonaws.com/myDeployment/searchCampusWire?term=precision

- Lambda Package: https://github.com/JosephineFalso/CourseProject/blob/main/lambdaPackages/searchCampusWire-1afbd7a5-d957-4450-aad6-48e0f66f2268.zip
- The API returns the JSON response. Sample response ->

```
{
  "dcg":[
    {
      "title":"Questions about Practice Quiz 3",
      "body":"Question 9 DCG is better than nDCG as its value is within [0 1].
      "url":"https://campuswire.com/c/G0A3AA370/feed/post/317",
      "rank":"1"
    },
    {
      "title":"IdealDCG",
      "body":"Why the idealDCG still has one document rated \"2\" instead of all 10 documents rated \"3\"?
![image.png](https|//campuspro-uploads.s3.us-west-2.amazonaws.com/0a3aa370-c917-4993-b5cf-
4e06585e7704/cd9732e7-9cdb-4d19-a06a-d431ec9550e1/image.png)",
      "url":"https://campuswire.com/c/G0A3AA370/feed/post/1125",
      "rank":"2"
    }
  ]
}
```

ii) searchPDFDoc -> Returns the link to the pdf and page numbers where the keyword is available.

- Endpoint: https://ki5xlwpsm9.execute-api.us-east-2.amazonaws.com/pdfdeployment/searchPDFDoc?term={keyword}
- Replace keyword with the actual keyword such as precision. For example, https://ki5xlwpsm9.execute-api.us-east-2.amazonaws.com/pdfdeployment/searchPDFDoc?term=precision
- Lambda Package: https://github.com/JosephineFalso/CourseProject/blob/main/lambdaPackages/searchPDFDoc-c967aa21-efc7-49ca-8247-804460e22605.zip
- The API returns the JSON response. Sample response ->

```
{
  "precision":[
    {
      "url":"https://i-share-
uiu.primo.exlibrisgroup.com/permalink/01CARLI_UIU/1ubbi2j/alma99839869412205899",
      "pagenumbers":"139,191-202,205,209-210,255,293,334,345-346,405,510-511,515,518-519,522,525"
    }
  ]
}
```

iii) getKeywords-> Returns the list of keywords for a particular week.

- Endpoint: https://312r0suowe.execute-api.us-east-2.amazonaws.com/keywords-stage/getkeywords?week={weeknumber}

- Replace the weeknumber with the actual week number. For example, https://312r0suowe.execute-api.us-east-2.amazonaws.com/keywords-stage/getkeywords?week=3
- Lambda Package: https://github.com/JosephineFalso/CourseProject/blob/main/lambdaPackages/getkeywords-0a96efcb-99b5-4652-a822-1e73d80bb252.zip
- Sample response ->

```
{
  "keywords":[
    [
      "precision"
    ],
    [
      "recall"
    ]
  ]
}
```

User Interface

The application UI is developed using reactjs. The application skeleton is created using the default way of creating a react app described on its website: **https://reactjs.org/docs/create-a-new-react-app.html.** Functional react components are mainly used in this application. Ableplayer which is a custom javascript video library is integrated with the app to play the videos and transcripts in sync. The videos and transcripts are placed in the public folder of the code base. A total of 4 main functional components are written along with ableplayer.js module downloaded from internet. You can get additional dependencies by looking into the package.json file.

The 4 functional components written are used in the App.js to render the content. The first component ProjectMenu.js will generate a mapping between Lecture Videos and their associated transcripts. It will then create a Menu displaying the Lecture names as submenu items.

When a user clicks on any of the submenu lecture entries, it sends the Lecture Video and Transcript information to the next functional component, i.e Custom.js. This component takes the input and starts the video and transcript player playing the provided video and transcript.

The third component displays additional info related to the keywords that are generated by the backend system. The component Keywords.js will make a call to back end api to get the keywords and their associated entries in the ***Campuswire*** feed and ***Course textbook*** present in university library. ***Axios*** library is used to make calls to the api's and generate the data. Note: Javascript is asynchronous, hence promises are used to resolve the api calls response.

The last component we have is RenderInWindow.js which opens a new tab using react inbuilt functionality called as ***createPortal***. Whenever a user clicks on the keywords that are displayed under menu to the right, a new tab will be opened with additional info and references related to the keyword.

Software Usage

API Deployment Steps

Prerequisites:

- Setup AWS account using the UIUC email address.
- Download the lambda package from the GitHub(lambda package URLs are provided for each of the APIs in the ***"API specifications"*** section.

Steps:

1)Login to AWS account and search for lambda service. Click on Functions and create a new function. Choose Python 3.8 as a runtime, enter function name and leave the rest of the fields with default values and click on create. The basic python function will be created.

2) Click on "Code" tab and click on Upload from .zip file. Choose the lambda package and upload the package. Click on "Deploy". This will deploy the lambda function.

3) Now, click on "Add Trigger" in the "Function Overview" section and choose "API Gateway" as a trigger. Choose "Open" as an option since there is no token or permissions required to access the API.

4) Click on the API Gateway once the trigger is added. On the API Gateway screen, add the Query String parameter named "term" for searchCampusWire and searchPDFDoc APIs or "week" for getKeywords API and mark it as required.

5) Finally, enable the CORs settings, so the API can be invoked from any domain.

TextInfo Web UI

The project is a reactjs app which has a catalog of CS 410 Text information Systems videos and their corresponding transcripts. The project makes use of a video player which plays the video and its associated transcript in sync. Users can navigate to any part of the video by clicking on the transcript. Along with that, the UI displays additional related information about course topics.The project makes use of backend api to fetch keywords from the transcripts and to provide additional reading material about the keywords.

Prerequisites for the app to deploy on local

Install nodejs using the link: https://nodejs.org/en/download/

To download all the application related dependencies.You can find the dependecies in the package.json file.

npm install

Runs the app in the development mode. Open http://localhost:3000 to view it in the browser.

npm start

The page will reload if you make edits. You will also see any lint errors in the console.

From the website, the user selects a lesson from the menu on the left. The user can play the Coursera lecture videos and read the transcription text from this website. A text box on the right of the screen displays the significant keyword topics from the video. Links to relevant CampusWire postings and page number references in the course textbook are provided here.

Software Usage Tutorial

A software usage tutorial video is located at:
https://uofi.box.com/s/be3qzaou0674238672u74suoh1dkhmdx

Team Members

The workload was distributed equally among all team members.

Josephine Falso

- Attempted Web Scraping of transcription text in course videos of CS410 lectures and Text Mining of keywords

Sree Alaparthi

- Text Mining of CampusWire postings and course textbook to identify additional information for each keyword
- API development

Narasimha Kethireddy

- UI development to display course videos and the corresponding transcripts