cumulus

project-2 / wdi-hamilton / david azaria / december 2017

what did i set out to do?

build on my interests and passions...

build on my interests and passions...

news

web development + data science

uphill challenges

cumulus

what am i proud of?

technical wins include

- multiple CRUD features
- successful API call to third party source
- lots of new (to me) npm package implementations
- MVC all up and running
- even utilized a `for loop` to unblock my progress
- lots of data manipulation, including but not limited to:
    - stringificaton
    - tokenization
    - stemming
    - object/array/string parsing and conversion

what was challenging?

technical challenges include

- implementation of the tfidf did not go as planned
- implementation of the word cloud did not go as planned
- a never-ending collection of small bugs, which i should have done a better job tracking
- limited css / postgres development
- heroku deployment could have gone smoother

all of this

cumulus

```
/* my external and internal dependencies rest here */
require('dotenv').config();
const express = require('express');

const cumulusRouter = express.Router();

/* axios is my api calling package */
const axios = require('axios');

/* natural is my nlp package that will help with my stemmer and as intended the tfidf */
const natural = require('natural');

/* this is my stopwords package which will help me, hopefully, remove words like the, and, is, or, etc. */
const sw = require('stopword');

/* this is my tokenizer which will break up my array of sentences into atomic words */
const tokenizer = new natural.WordTokenizer();

/* here i bring in my searchdb model to allow me the ability to inject my searches into the db */
const searchDB = require('../models/searchesDB');

/* this views required will hopefully eventually be used to utilize ejs functionality of my api call render */
const views = require('../controllers/viewController.js');

/* hiding my api key so pirates cannot steal it from me! */
const apikey = process.env.APIKEY;

/* hit the API and get back an array of results */
function hitAxios(req, res, next) {
  axios.get('https://newsapi.org/v2/top-headlines', {
    params: {
      language: 'en',
      apikey,
    },
  })
  /* grab the articles from the response data */
    .then(({ data: { articles } }) => {
    /* run through 900 articles */
      const matrix = articles.slice(1, 901)
      /* map over and extract only the description from each article */
        .map(({ description, title }) => title + description);
      res.locals.matrix = matrix;
      next();
    })
    .catch(err => res.send(err));
  /* catch any errors from anything above */
}

/* this function tokenizes the data, which in this context means breaks up sentences into their atomic parts, in other words into words. */

function tokenizeData(req, res, next) {
  /* here i set a new variable which will be the result of an accumulation function returning the concatination of of reduced words into a blank array */
  const allTokenized = res.locals.matrix.reduce((acc, val) => {
    return acc.concat(tokenizer.tokenize(val));
  }, []);

  /* here i reassign a res.locals so that i can then stash and passalong the dataset to a new function below */

  res.locals.allTokenized = allTokenized;
  next();
  debugger;
}

/* this function is a bit more than meets the eye since i want to not only stem words, but introduce a stopwords package before doing so. this might benefit from future refacotring to further separate my concerns. either way, in this function i stem my words and introduce a stopwords package */

function stemWords(req, res, next) {
  const oldTokenized = res.locals.allTokenized;
  const newTokenized = sw.removeStopwords(oldTokenized);
  res.locals.newTokenized = newTokenized;

  /* in this function, i follow a similar methodology to my tokenized data by running a accumulating function returning a conctatinated result into a blank array */

  const allStemmed = res.locals.newTokenized.reduce((acc, val) => {
    return acc.concat(natural.PorterStemmer.stem(val));
  }, []);
  /* again restoring my sanitized dataset into a res.locals to further sanitize elsewhere */
  res.locals.allStemmed = allStemmed;
  next();
  debugger;
}
```

```
const allStemmed = res.locals.newTokenized.reduce((acc, val) => {
  return acc.concat(natural.PorterStemmer.stem(val));
}, []);
/* again restoring my sanitized dataset into a res.locals to further sanitize elsewhere */
res.locals.allStemmed = allStemmed;
next();
debugger;
}

/* because i was experiencing issues implementing a tfidf that did not return back null values - more to discuss with jason seminara - i made the executive decision to simply sum words (tf) irrespective of their idf. */

function sumWords(req, res, next) {
  const wordArray = res.locals.allStemmed;
  /* because i knew i needed to at some point strip out just words after summing them, i decided to introduce an empty object which i will execute a loop into to sum particular words based on an if else condition i set up. from there i can then identify those words, and counts, if necessary, elsewhere in the project */
  const newObject = {};
  /* this for loop here sets up a fun if else statement. if the word in the array has been seen, increment its count. if it hasn't, set it as one so if/when it comes back, i can increment its tf */
  for (let i = 0, j = wordArray.length; i < j; i++) {
    if (newObject[wordArray[i]]) {
      newObject[wordArray[i]]++;
    } else {
      newObject[wordArray[i]] = 1;
    }
  }
  res.locals.sumWords = newObject;
  next();
  debugger;
}

/* this function will first sort by sum size and then slice the returned words by a measure i dictate */

function sortWords(req, res, next) {
  function sortObject(object) {
    /* i will use this empty array as my empty container for pushing into it object properties i deem appropriate, which in this case are the words and counts */
    const sortingArray = [];
    let property;
    for (property in object) {
      if (object.hasOwnProperty(property)) {
        sortingArray.push({
          word: property,
          count: object[property],
        });
      }
    }
    /* here i introduce my sorting array to display those with the highest counts first */
    sortingArray.sort((first, second) => {
      return second.count - first.count;
    });
    return sortingArray;
  }
  const list = res.locals.sumWords;
  const newarr = sortObject(list);
  /* here i specify that i am looking for the top 30 words based on how often they appear */
  const slicedArr = newarr.slice(1, 31);
  res.locals.sortedWords = slicedArr;
  next();
  debugger;
}

/* in this function i will attempt to stringify the words, taking them out of an array and using a join() method to break up its pieces and sending back a comma separated list of atomic words */
function stringifyWords(req, res, next) {
  /* here i am reassigning my res.locals.sortedWords object to a const i will then handle with a stringify method */
  const toDisplay = res.locals.sortedWords;
  const words = toDisplay.map((stringify) => {
    /* here i am stringifying through a .map function the word property of my words object */
    return stringify.word;
  });
  /* the .join() method here allows me to comma separate and serialize the words after stringifying the words out of their original array */
  const newWords = words.join(', ');
  res.locals.words = newWords;
  next();
  debugger;
}

/* this function injects each result string into the database */
function injectSave(req, res, next) {
  searchDB.save(res.locals.words)
    .then(() => {
      next();
      debugger;
    })
    .catch(err => next(err));
```

was humbling

cumulus

our results

here our words will appear from most common to least, singling out the top thirty words currently found in english-speaking news headlines across the world

trump, presid, court, will, new, sai, year, travel, ban, suprem, mondai, t, tax, not, moor, donald, nation, mai, allow, senat, roi, back, on, U, it, republican, S, time, hous, two

hit api results again see all past cumulus api searches go back to homepage

cumulus

questions? comments? concerns?