CSE 6331 Cloud Computing
Spring 2021, © DL, UTA, 2021

Programming Assignment 6
Either Google Cloud or Amazon AWS

Cloud Assignment 6 – dealing with "large" textual information

Description:
Much of the available, digitized information is in the form of text.
(We will consider web, and similar, as a variation of text.) We will simplify handling to only text, but the concepts could be extended.
More:
Given several (say a few hundred or thousand) text files, which we will generically call documents, we want to find those documents relevant to a user's needs (requests.)
(This is, in general, what "search engines" do.)
For this assignment we will simplify many of the interesting details, but try to emphasize many of the issues and interesting approaches to searching text. Almost all original, source documents need to be "cleaned" (may include removing pictures, font details, pagination, and similar.)
Here ALL non ASCII information will be removed.
(Note, we will assume that docs are in English, but Spanish, French and similar languages are not difficult to extend processing, Chinese is more difficult.)
Doing a simple word scan is simple, very time consuming, and usually gives poor results. Preprocessing the original docs is important.
The following are simple original doc processing:
Dealing with upper or lower case letters, usually changing to lower case. Remove punctuation (or most)
Remove very common words ("stop words" such as the, or, and.)
Additionally (optionally):
Word stemming (cats -> cat)
(Many others)

Then the words in the documents are extracted and indexed, with "pointers" back to the individual relevant documents, and where found.

Then, through some sort of interface, a "search" of relevant documents can be done.
The simplest, and usually poorest result is a single word, such as "cloud".
Combinations of words, in close proximity, usually do better "cloud computing".
(These are sometimes referred to as bi-grams, and tri-grams, and extensions.)
One, free, source of thousands of texts is: https://www.gutenberg.org/
Notice that small optimizations will often give much better results:
For example "red hot" may be the same as "hot red" (interchange word order) Letters in search may be transposed (or missing): "teh" instead of the, questionble instead of questionable
There are word lists that may help: (MIT site, github, many others, depending on what you want.)

Users of this service will interact with your service through web page interfaces, all processing and web service hosting is (of course) cloud based.
Additional Details:
A user should be able to do searches based on words or word combinations to find
Relevant documents and where in that document (such as line or offset.)
(Show lines or paragraphs that match.)
~~Please, submit through Canvas.~~
~~All work must be your own, or from a group (Same as previous assignments)~~

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <style>
      table,
      td {
        border: 1px solid black;
      }
    </style>
    <title>Assignment 1</title>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link type="text/css" rel="stylesheet" href="css.css" />
    <link type="css" rel="stylesheet" href="style.css" />
    <script type="text/javascript" src="assignment.js"></script>
  </head>
  <body>
    <div class="header">
      <h1>Sahithi Edupuganti</h1>
      <p>UTA ID: 1001759482</p>
    </div>

    <div class="mb-10">
      <label>Word</label>
      <input
        type="text"
        id="search_word"
        placeholder="Enter "
        name="search_word"
      />
      <label>File Name</label>
      <input
        type="text"
        id="file_name"
        placeholder="Enter "
        name="file_name"
      />
      <button type="button" onclick="show_twenty_lines('file_name', 'search_word')">Show</button>

      <button type="button" onclick="showWordCount()">Show Word Count</button>
      <button type="button" onclick="showFirstThreeLines()">Show First Three Lines</button>
    </div>

    <table style="width: 100%"></table>
    <div id="output_div">
      <div id="myData" style="color: green"></div>
      <div id="myDataNo"></div>
    </div>
    <div id="wordCount">
    </div>
    <div id="threeLines">
    </div>
  </body>
```

```
</html>
```

```
function preprocess() {
  fetch("/preprocess", {
    method: "GET", // or 'PUT'
    headers: {
      "Content-Type": "application/json",
    },
  })
    .then((response) => response.json())
    .then((data) => {})
    .catch((error) => {
      console.error("Error:", error);
    });
  return false;
}

function showWordCount() {
  fetch("/showWordCount", {
    method: "GET", // or 'PUT'
    headers: {
      "Content-Type": "application/json",
    },
  })
    .then((response) => response.json())
    .then((data) => {
      add_word_count_table(data.data);
    })
    .catch((error) => {
      console.error("Error:", error);
    });
  return false;
}

function showFirstThreeLines() {
  fetch("/showFirstThreeLines", {
    method: "GET", // or 'PUT'
    headers: {
      "Content-Type": "application/json",
    },
  })
    .then((response) => response.json())
    .then((data) => {
      add_first_three_lines_table(data.data);
    })
    .catch((error) => {
      console.error("Error:", error);
    });
  return false;
}
```

```
function search(search_word) {
  var search_word = document.getElementById(search_word).value;
  fetch("/search", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify({ search_word: search_word }),
  })
    .then((response) => response.json())
    .then((data) => {
      console.log("Success:", data);
      add_data_to_table(data.data);
    })
    .catch((error) => {
      console.error("Error:", error);
    });
  return false;
}

function show_twenty_lines(file_name, search_word) {
  var file_name = document.getElementById(file_name).value;
  var search_word = document.getElementById(search_word).value;
  fetch("/showLines", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify({ file_name: file_name, search_word: search_word }),
  })
    .then((response) => response.json())
    .then((data) => {
      console.log("Success:", data);
      add_twenty_lines(data.data);
    })
    .catch((error) => {
      console.error("Error:", error);
    });
  return false;
}

function add_data_to_table(data) {
  var headers = ["name", "line_number", "line"];
  var headings = ["File Name", "Line Number", "Sentence"];

  document.getElementById("output_div").innerHTML = "";

  var outputHTML = "";
  outputHTML += "<table>";
  outputHTML += "<tr>";
  for (var i = 0; i < headers.length; i++) {
    outputHTML += "<th>" + headings[i] + "</th>";
  }
  outputHTML += "</tr>";
  for (var i = 0; i < data.length; i++) {
    outputHTML += "<tr>";
    for (var j = 0; j < headers.length; j++) {
      outputHTML += "<td>" + data[i][headers[j]] + "</td>";
    }
    outputHTML += "</tr>";
  }
  outputHTML += "</table>";
  // output our html
  document.getElementById("output_div").innerHTML = outputHTML;
}

function add_word_count_table(data) {
  var headers = ["name", "word_count"];
  var headings = ["File Name", "Word Count"];

  document.getElementById("wordCount").innerHTML = "";

  var outputHTML = "";
  outputHTML += "<table>";
  outputHTML += "<tr>";
  for (var i = 0; i < headers.length; i++) {
    outputHTML += "<th>" + headings[i] + "</th>";
  }
  outputHTML += "</tr>";
  for (var i in data) {
    outputHTML += "<tr>";
    outputHTML += "<td>" + i + "</td>";
    outputHTML += "<td>" + data[i] + "</td>";
    outputHTML += "</tr>";
  }
  outputHTML += "</table>";
  // output our html
  document.getElementById("wordCount").innerHTML = outputHTML;
}

function add_first_three_lines_table(data) {
  var headers = ["name", "three_lines"];
  var headings = ["File Name", "1st Line", "2nd Line", "3rd Line"];

  document.getElementById("wordCount").innerHTML = "";

  var outputHTML = "";
  outputHTML += "<table>";
  outputHTML += "<tr>";
  for (var i = 0; i < headings.length; i++) {
    outputHTML += "<th>" + headings[i] + "</th>";
  }
  outputHTML += "</tr>";
  for (var i in data) {
    outputHTML += "<tr>";
    outputHTML += "<td>" + i + "</td>";
    outputHTML += "<td>" + data[i][0] + "</td>";
    outputHTML += "<td>" + data[i][1] + "</td>";
    outputHTML += "<td>" + data[i][2] + "</td>";
    outputHTML += "</tr>";
  }
  outputHTML += "</table>";
  // output our html
  document.getElementById("wordCount").innerHTML = outputHTML;
}


function add_twenty_lines(data) {
  var headers = ["name", "three_lines"];
  var headings = ["Count", "Lines"];

  document.getElementById("wordCount").innerHTML = "";

  var outputHTML = "";
  outputHTML += "<table>";
  outputHTML += "<tr>";
  for (var i = 0; i < headings.length; i++) {
    outputHTML += "<th>" + headings[i] + "</th>";
  }
  outputHTML += "</tr>";

  outputHTML += "<tr>";
  outputHTML += "<td>" + data.count + "</td>";
  outputHTML += "<td>" + data.lines.join('\n') + "</td>";
  outputHTML += "</tr>";

  outputHTML += "</table>";
  // output our html
  document.getElementById("wordCount").innerHTML = outputHTML;
}



const express = require("express");
const router = express.Router();
//const routes = require("./routes/home");
const http = require("http");
var fs = require('fs');
var app = express();
var LineByLineReader = require("line-by-line");
var stopword = require("stopword");
const removePunctuation = require("remove-punctuation");
app.use(express.static("./routes"));

const bodyParser = require('body-parser');

app.use(bodyParser.urlencoded({ extended: false }));

// parse application/json
```

```javascript
app.use(bodyParser.json());

var words_file_line = {};
var file_line = {};
var stopwords = [];

var files = ["A1.txt","A2.txt","A3.txt", "C1.txt","C2.txt", "S1.txt", "S2.txt", "S3.txt",
"S4.txt", "S5.txt"];
var file_words = {};
var first_three_lines = {};

function getMapValue(map, key) {
  return map[key] || [];
}

async function loadStopWords() {
  lr = new LineByLineReader('shortliststopwords.txt');
  lr.on("line", function (line) {
    stopwords.push(line.replace(/"/g, '').trim());
  });

  lr.on("end", function () {
    console.log('Completed Loading stop words');
    console.log(stopwords);
  });
}

function indexFile(num, word_count) {
  var file_index = num.toString();
  var line_no = 1;
  var array_sw = [];

  lr = new LineByLineReader(files[num]);

  lr.on("line", function (line) {
    word_count = word_count + line.split(/\s+/).length;
    const words = stopword.removeStopwords(
      removePunctuation(line).split(/\s+/), stopwords
    );

    for (var i = 0; i < words.length; i++) {
      const word = words[i].toLowerCase();
      words_file_line[word] = getMapValue(words_file_line, word).concat(
        file_index + "_" + line_no
      );
    }
    file_line[file_index + "_" + line_no] = words.join(' ');

    if(line_no <= 3){
      first_three_lines[files[num]] = getMapValue(first_three_lines,
files[num]).concat(words.join(' '));
    }
    line_no++;

  });

  lr.on("end", function () {
    //console.log('Completed Indexing file ' + num.toString());
    file_words[files[num]] = word_count;
  });
}

async function indexFiles() {
  await loadStopWords();
  for (var num = 0; num < files.length; num++) {
    var word_count = await indexFile(num, 0);
  }
}

router.get("/", function (req, res) {
  res.sendFile("index.html", { root: __dirname });
});

router.get("/preprocess", (req, res) => {
  indexFiles();
  res.json({message: 'Done preprocessing'});
});

router.get("/showWordCount", (req, res) => {
  res.json({data: file_words});
});

router.get("/showFirstThreeLines", (req, res) => {
  res.json({data: first_three_lines});
});

router.post("/showLines", (req, res) => {
  var file_name = req.body.file_name;
  var word = req.body.search_word.toLowerCase();

  var file_index;
  var lines = [];

  for(var i = 0; i < files.length; i++){
    if(files[i] === file_name)
      file_index = i;
  }

  for(var i = 0; i < words_file_line[word].length; i++){
    var [index, line] = words_file_line[word][i].split('_');
    var count = 0;
    index = parseInt(index);
    if(index == file_index){
      count++;
      if(count <= 20){
        lines.push(file_line[words_file_line[word][i]]);
      }
    }
  }

  res.json({data: {count: count, lines: lines}});
});



router.post("/search", (req, res) => {
  var inp = req.body.search_word;
  var inp = inp.toLowerCase();
  var out = [];
  var array = inp.split(/\s+/);

  var response = [];

  for (var i = 0; i < array.length; i++) {
    if(words_file_line[array[i]]){
      for(var j = 0; j < words_file_line[array[i]].length; j++){
        var [index, line] = words_file_line[array[i]][j].split('_');
        const line_number = parseInt(line);
        response.push({name: files[parseInt(index)], line_number: line_number, line:
file_line[words_file_line[array[i]][j]]});
      }
    }
  }

  res.json({data: response});
});



module.exports = router;

app.use("/", router);
app.listen(8080);
```