ITE 318

Open Source Programming

A2 Slot

Text Editor based on Electron

Submitted To: Submitted By:

Prof. Pradeep Kumar Reddy Saurabh Mathur - 14BIT0180

Abishek Aditya - 14BIT0137

Ruchir Bhardwaj - 14BIT0195

Pulkit Narwani - 14BIT0179

# Project Overview

1. Idea

A plain text editor with spell checking functionality (highlights the spelling errors) which also provide user with the option to save the text he has written.

2. Functionality

*Spell Checking -* Highlights the wrong word which is checked against probability of the wod being correct. The algorithm is that for every string, four simple edits are created. A simple edit to a word is a deletion (remove one letter), a transposition (swap two adjacent letters), a replacement (change one letter to another) or an insertion (add a letter). These edits are returned as a set of edited strings (whether or not words can be made). So, for a word of length *n*, there will be *n* deletions, *n*-1 transpositions, 26*n* alterations, and 26(*n*+1) insertions, for a total of 54*n*+25 (of which a few are typically duplicates); which can be a very large number. This number can be reduced if we restrict the set to known words in the dictionary.

*Save -* The text value is passed from the html page as a document object which is later appended to file variable.

**Technologies Used**

1. Electron Framework

Electron(formerly known as *Atom Shell*) is an open-source framework developed by [GitHub](https://en.wikipedia.org/wiki/GitHub). It allows for the development of desktop [GUI](https://en.wikipedia.org/wiki/GUI) applications using [front and back end](https://en.wikipedia.org/wiki/Front_and_back_end) components originally developed for [web applications](https://en.wikipedia.org/wiki/Web_application): [Node.js](https://en.wikipedia.org/wiki/Node.js) runtime for the backend and [Chromium](https://en.wikipedia.org/wiki/Chromium_(web_browser)) for the frontend. Electron is the main GUI framework behind several notable open-source projects including GitHub's [Atom](https://en.wikipedia.org/wiki/Atom_(text_editor)) and [Microsoft](https://en.wikipedia.org/wiki/Microsoft)'s [Visual Studio Code](https://en.wikipedia.org/wiki/Visual_Studio_Code) [source code editors](https://en.wikipedia.org/wiki/Source_code_editor) and the Light Table [IDE](https://en.wikipedia.org/wiki/Integrated_development_environment).

A basic Electron app consists of three files: *package.json* (metadata), *main.js* (code) and *index.html* ([graphical user interface](https://en.wikipedia.org/wiki/Graphical_user_interface)).

The most important file is package.json, since it keeps the information about the package viz. *“name”* – application name, *“version”* – the application version string and *“main”* – name of the main script file of the application.

The framework is provided by the Electron executable file (*electron.exe* in [Windows](https://en.wikipedia.org/wiki/Windows), *electron app* on [macOS](https://en.wikipedia.org/wiki/MacOS) and electron on [Linux](https://en.wikipedia.org/wiki/Linux)). Developers wishing to add branding and custom [icon](https://en.wikipedia.org/wiki/Icon_(computing)) can rename and/or edit the Electron executable file.

2. TypeScript Language

[TypeScript](http://www.typescriptlang.org/) is a strict superset of JavaScript which primarily provides optional static typing, classes and interfaces. One of the big benefits is to enable ides to provide a richer environment for spotting common errors as you type the code. It is a free and open source programming language developed and maintained by Microsoft.

**Source**

import { app, BrowserWindow, Menu, MenuItem } from 'electron';

let win;

let createWindow = () => {

//

// Actually open the window

win = new BrowserWindow({width:800, height: 900});

win.loadURL(`file://${\_\_dirname}/index.html`);

createMenu();

win.on('closed', () => {

win = null;

});

}

app.on('ready', createWindow);

app.on('window-all-closed', () => {

if(process.platform !== 'darwin') {

app.quit();

}

});

app.on('activate', () => {

if(win === null) createWindow();

});

let createMenu = () => {

console.log("Create Menu");

const menu = Menu.getApplicationMenu();

if (menu == null) return;

menu.insert(0,new MenuItem({

label: 'File',

submenu: [

{

label: 'Save',

click: () => {

win.webContents.send('save');

}

}

]

})

);

Menu.setApplicationMenu(menu);

}

**renderer.ts**

const dialog = require('electron').remote.dialog

var ipc= require('electron').ipcRenderer;

ipc.on('save',function(event){

var content=(<HTMLTextAreaElement>document.getElementById("textarea")).value;

var fs = require('fs');

// Or with ECMAScript 6

const {dialog} = require('electron').remote;

// You can obviously give a direct path without use the dialog (C:/Program Files/path/myfileexample.txt)

dialog.showSaveDialog((fileName) => {

if (fileName === undefined){

console.log("You didn't save the file");

return;

}

// fileName is a string that contains the path and filename created in the save file dialog.

fs.writeFile(fileName, content, (err) => {

if(err){

alert("An error ocurred creating the file "+ err.message)

}

alert("The file has been succesfully saved");

});

});

});

**mine.js**

var $container = $('.container');

var $backdrop = $('.backdrop');

var $highlights = $('.highlights');

var $textarea = $('textarea');

var $toggle = $('button');

var $words = Object.keys(data);

var $letters = 'abcdefghijklmnopqrstuvwxyz'.split('');

var $corrector = new SpellCorrector($letters,data);

$(".head").empty();

function applyHighlights(text) {

text = text

.replace(/\n$/g, '\n\n');

var $textarr = text.trim().split(/[ ]/);

$(".head").empty();

$.each($textarr, function (i, val) {

var $val = val.replace(/[\.\,;\:]/, "");

var $seen = {};

if($val.startsWith("'")){

var $pattern = new RegExp("'" + $val.replace(/'/g,"") + "'", "g" );

}

else if($val.startsWith('"')){

var $pattern = new RegExp('"' + $val.replace(/"/g,"") + '"', "g" );

}

else

{

var $pattern = new RegExp("\\b" + $val + "\\b" , "g");

}

if ($.inArray($val.toLowerCase().replace(/"/g,"").replace(/'/g,''), $words) == -1) {

text = text.replace($pattern, '<mark>' + $val + '</mark>');

if($val != " "){

if($val.startsWith("'")){

console.log($.trim($val));

var $btn = "<button onClick=correct(token)>" + $val + "</button>";

$btn = $btn.replace('token', '"'+ $val + '"');

$button = $($btn);

}else{

var $button = $('<button onClick="correct(\''+ $.trim(val) +'\')">' + $val + '</button>');

}

$(".head").append($button);

$("button").each(function() {

var $txt = $(this).text();

if ($seen[$txt]){

$(this).remove();

} else {

$seen[$txt] = true;

}

});

}

}

});

return text;

}

function handleInput() {

var text = $textarea.val();

var highlightedText = applyHighlights(text);

$highlights.html(highlightedText);

}

function handleScroll() {

var scrollTop = $textarea.scrollTop();

$backdrop.scrollTop(scrollTop);

var scrollLeft = $textarea.scrollLeft();

$backdrop.scrollLeft(scrollLeft);

}

function bindEvents() {

$textarea.on({

'input': handleInput,

'scroll': handleScroll

});

}

function correct(str){

var $corrected = $corrector.correct(str.replace(/'/g,''));

$textarea.val($textarea.val().replace(str,$corrected));

handleInput();

$('#textarea').focus();

}

bindEvents();

handleInput();

console.log($corrector.correct('speling'));

**spelling.ts**

interface Counter {

[key: string]: number;

}

class SpellCorrector {

private \_wordCount: Counter;

private \_nWords: number;

private readonly \_letters: string[];

constructor(letters: string[],data: Counter) {

this.\_letters = letters;

this.\_wordCount = data;

this.\_nWords = Object.keys(this.\_wordCount).length;

}

correct(word): string {

if (this.\_wordCount.hasOwnProperty(word)) return word;

const validWords: string[] = this.\_computeEdits2(word)

.filter(w => this.\_wordCount.hasOwnProperty(w));

const probabilities = validWords.map(w => (this.\_wordCount[w] + 1) / this.\_nWords );

const index = this.\_indexOfMaximum(probabilities);

return validWords[index];

}

private \_indexOfMaximum(list: number[]) {

return list.reduce(

(bestIndexSoFar, currentlyTestedValue, currentlyTestedIndex) =>

currentlyTestedValue > list[bestIndexSoFar] ? currentlyTestedIndex : bestIndexSoFar, 0);

}

private \_computeEdits1(word: string): string[] {

let edits: string[] = [];

const addToEdits = (candidate) => (edits.indexOf(candidate) === -1) ? edits.push(candidate): null;

// delete

for (let i = 0; i < word.length; i++) {

addToEdits(word.slice(0, i) + word.slice(i + 1));

}

// transpose

for (let i = 0; i < word.length - 1; i++) {

addToEdits(word.slice(0, i) + word.slice(i + 1, i + 2) + word.slice(i, i + 1) + word.slice(i + 2));

}

// alter

for (let i = 0; i < word.length; i++) {

this.\_letters.forEach(l => addToEdits(word.slice(0, i) + l + word.slice(i + 1)));

}

// insert

for (let i = 0; i <= word.length; i++) {

this.\_letters.forEach(l => addToEdits(word.slice(0, i) + l + word.slice(i)));

}

return edits;

}

\_computeEdits2(word: string): string[] {

return this.\_computeEdits1(word)

.map(edit => this.\_computeEdits1(edit))

.reduce((previous: string[], current: string[]) => {

return previous.concat(current);

}, []);

}

}

/\*

var letters = 'abcdefghijklmnopqrstuvwxyz'.split('');

var corrector = new SpellCorrector(letters);

corrector.loadCounts('word\_counts.json', (err) => {

if (err) throw err;

console.log(corrector.correct('speling'));

});

\*/

**Steps To Run**

1. Install node.js
2. Install electron
3. Inside text editor folder run command on terminal - “npm start”

**Output**

