

Author(s)

Step 5 of 8



Note: Though this lab is split into multiple parts, it is highly recommended it is all completed in one sitting as the project environment may not be persisted. If for any reason that is not possible, please push your changes to git so that the changes are copied to your remote repository and you can start where you left.

If you need to refresh your memory on how to commit and push to GitHub in Theia lab environment, please refer to this lab [Working with git in the Theia lab environment](#)

Final Project (~3 hours)

Part A: Fork the Git repository to have the server and client code you need to start

1. Go to the [repository](#) which has the partially developed code for server side and client side.
2. Create a fork of the repository into your own. You will need to have a github account of your own to do so.

ibm-developer-skills-network / cazgi-IBM-Wats

generated from [ibm-developer-skills-network/coding-project-template](#)

[<> Code](#)[! Issues](#)[🔗 Pull requests](#) **3**[▶ Actions](#)[🔗 master ▼](#)[🔗 4 branches](#)[🏷️ 0 tags](#)

LavanyaTS Changed directory name



sentimentAnalyzeClient

Change



sentimentAnalyzeServer

Create n



.DS_Store

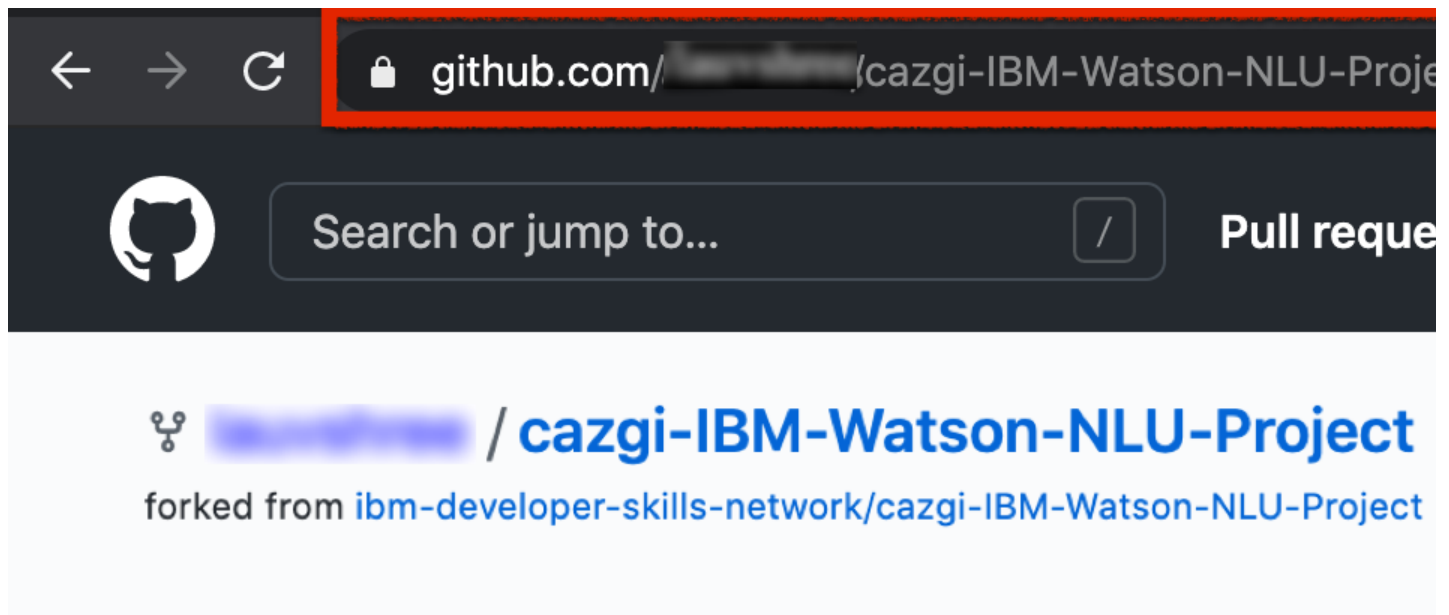
Change



.gitignore

adding v

3. Copy the **url** of your repository. **This needs to be submitted as part of the final submission.**




4. Go to your repository and copy the clone url.








[Code](#) [Pull requests](#) [Actions](#) [Projects](#)

master ▾

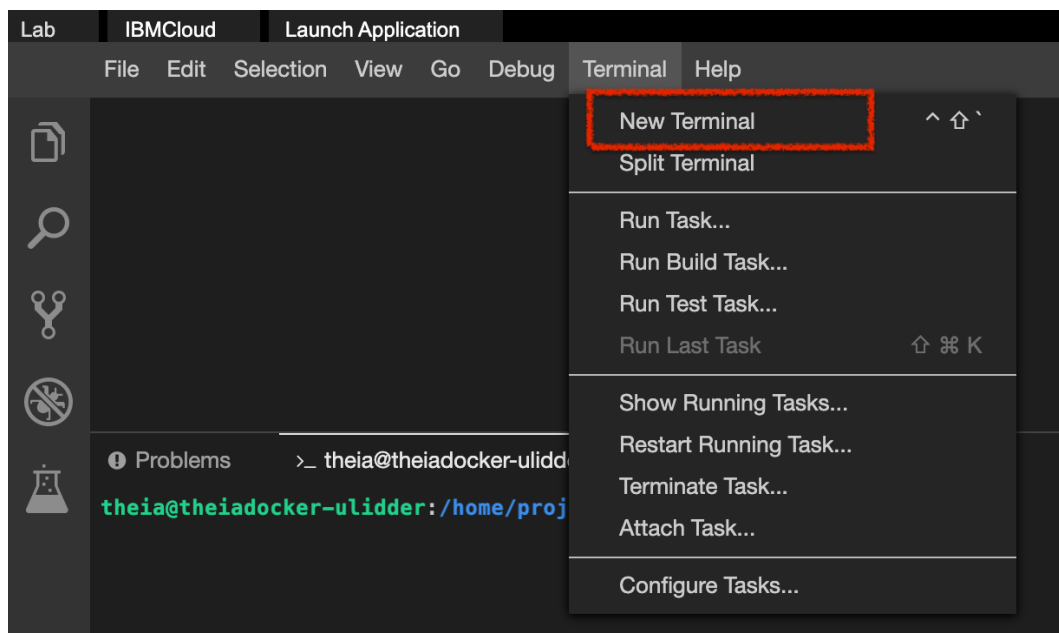
 4 branches 0 tags

This branch is even with ibm-developer-skills-network:ma

 **LavanyaTS** Changed directory name

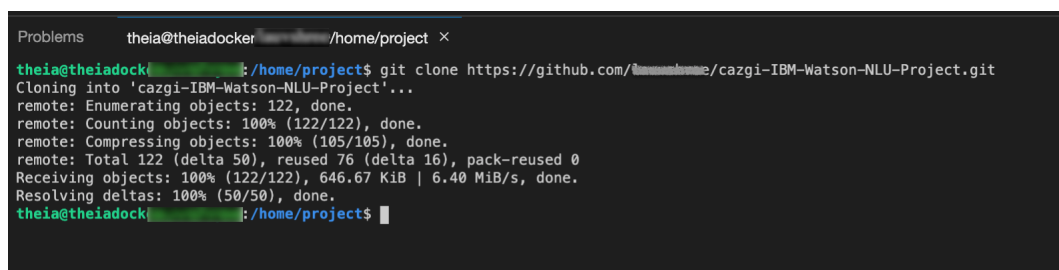
	sentimentAnalyzeClient	Changed dir
	sentimentAnalyzeServer	Create mani
	.DS_Store	Changed the
	.gitignore	adding vsco
	LICENSE	Initial comm
	README.md	Initial comm
	manifest.yml	updating ma

5. Open a terminal.



6. Clone the repository by pasting the url you copied from the git repository.

```
git clone <your_repo_name>
```



7. This will clone the repository with server and client files in your home directory in the lab environment. Check the same with the following command.

```
ls
```

```
{: codeblock}
```

The output should list the directory named `cazgi-IBM-Watson-NLU-Project`.

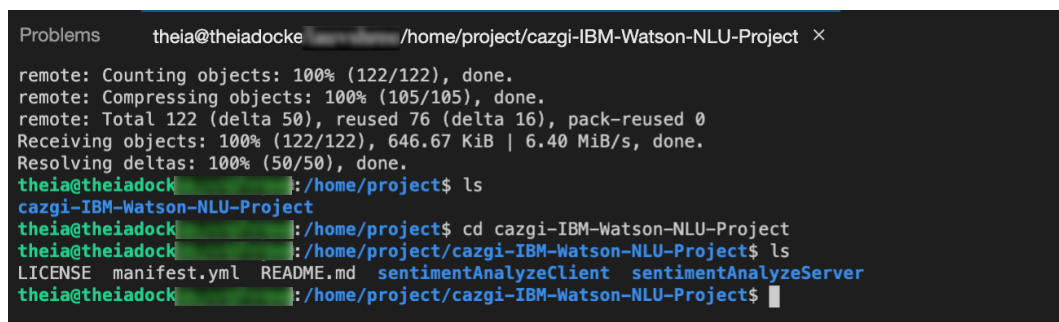
8. Run the following command to update npm version. `npm install npm@latest -s`

```
{: codeblock}
```

9. Change to the project directory and check its contents.

```
cd cazgi-IBM-Watson-NLU-Project && ls
```

```
{: codeblock}
```



Part B: Install Server-side packages

1. Change to the server directory. `cd /home/project/cazgi-IBM-Watson-NLU-Project/sentimentAnalyzeServer`

{: codeblock}

2. All packages are needed to be installed are listed in `package.json`. Run `npm install -s`, to install and save those packages.

```
npm install -s
```

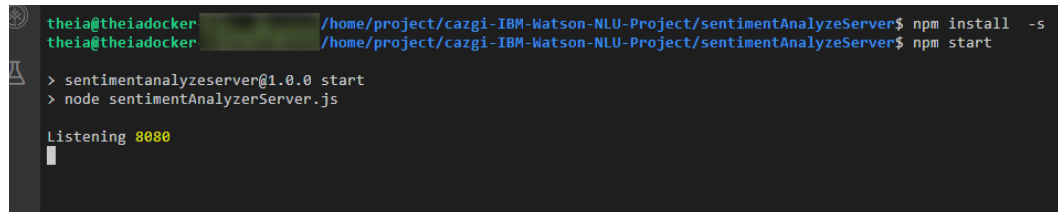
{: codeblock}

-  Click here for the checklist of required packages

3. Run the server using the following command. `npm start`

{: codeblock}

4. If all the required packages are successfully installed, the server should start without any issues.



```
theia@theiadocker: /home/project/cazgi-IBM-Watson-NLU-Project/sentimentAnalyzeServer$ npm install -s
theia@theiadocker: /home/project/cazgi-IBM-Watson-NLU-Project/sentimentAnalyzeServer$ npm start

> sentimentanalyzeserver@1.0.0 start
> node sentimentAnalyzerServer.js

Listening 8080
```

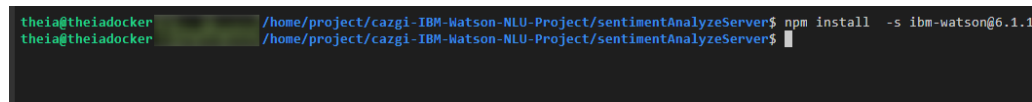
5. Press `Ctrl+C` to stop the server.

Part C: Install IBM Watson package and set up the `.env` file

1. Install the `ibm-watson` package in your server side using the following command.

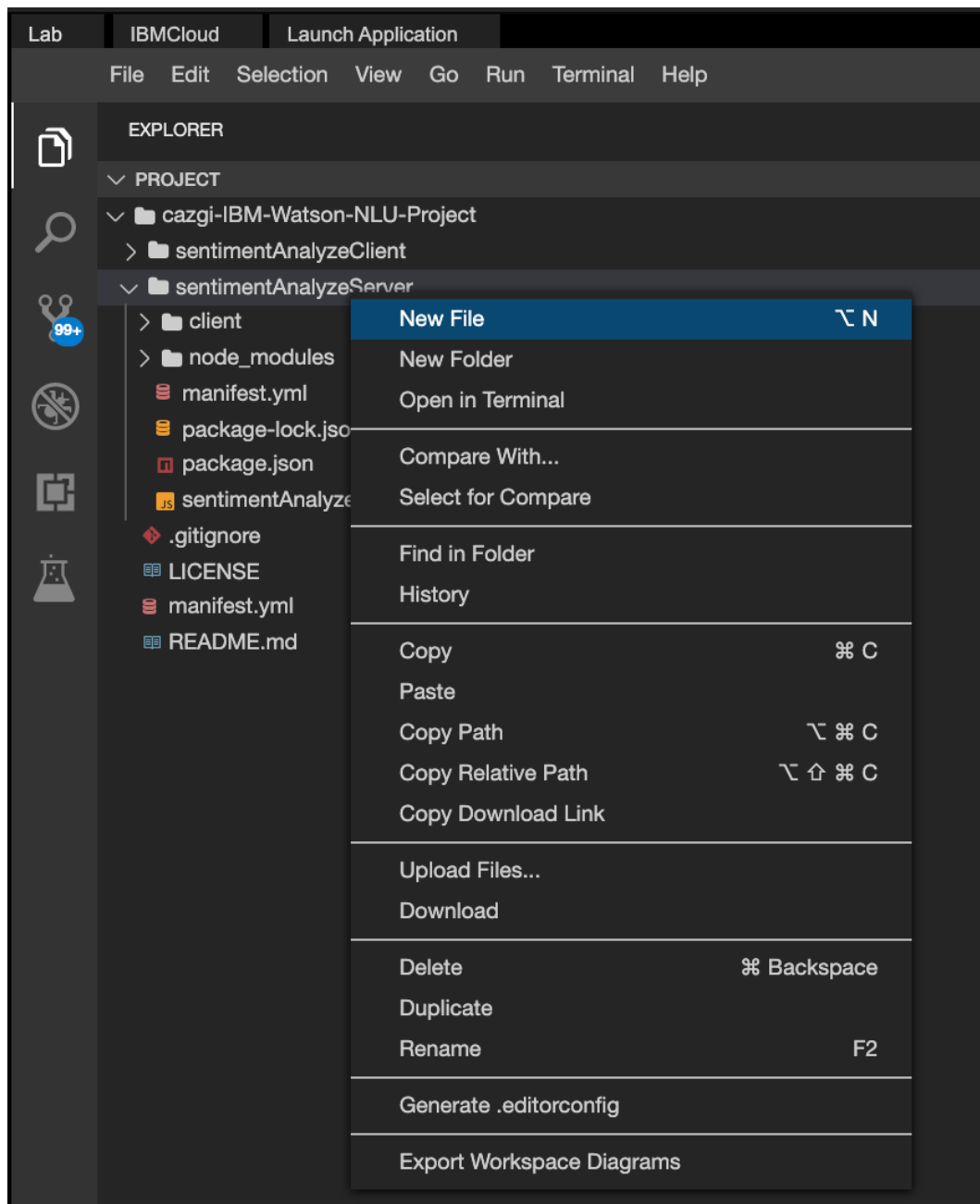
```
npm install -s ibm-watson@6.1.1
```

{: codeblock}

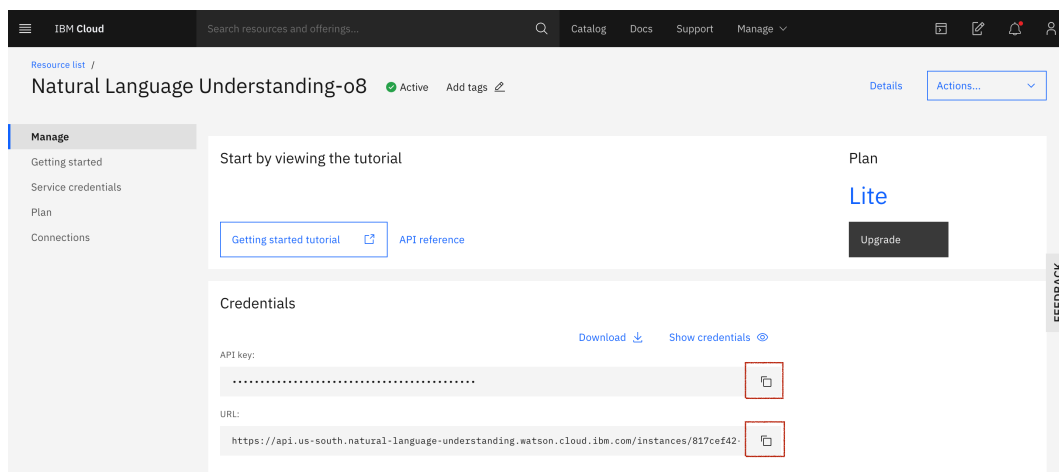


```
theia@theiadocker: /home/project/cazgi-IBM-Watson-NLU-Project/sentimentAnalyzeServer$ npm install -s ibm-watson@6.1.1
theia@theiadocker: /home/project/cazgi-IBM-Watson-NLU-Project/sentimentAnalyzeServer$
```

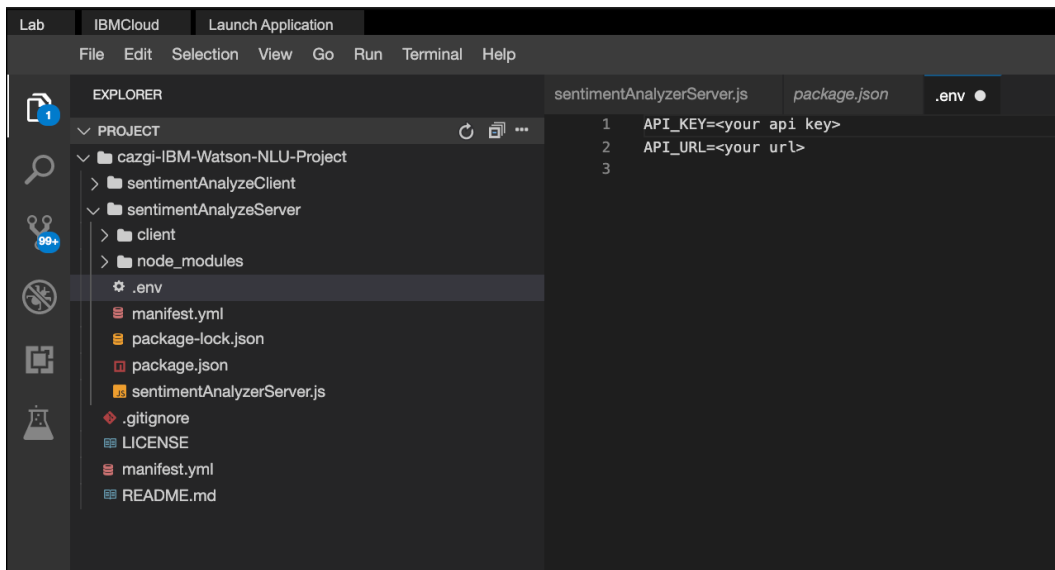
2. Right click on the `sentimentAnalyzeServer` in the explorer and a new file named `.env`.



3. Copy the credentials to point to your Watson NLU credentials from the IBM cloud, if you have not already done so in the previous lab.



4. Add the credentials required to the .env file you created.



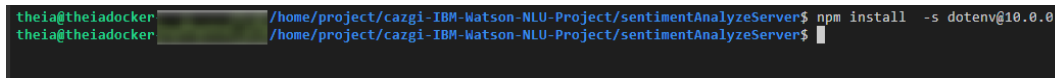
5. The contents of the .env file would be as below

```
API_KEY=<your api key>
API_URL=<your url>
```

6. Install dotenv to use the .env file in the server application using the following command.

```
npm install -s dotenv@10.0.0
```

{: codeblock}



7. Make changes in your express JS (in the file named sentimentAnalyzerServer.js) to define and implement a method `getNLUInstance()` where you create an instance of `NaturalLanguageUnderstanding` using the credential from the .env file using `dotenv` package and return the instance. You can refer to this [link](#) for more details on how to do this.

```
const NaturalLanguageUnderstandingV1 = require('ibm-watson/natural-language-understanding/v1');
const { IamAuthenticator } = require('ibm-watson/auth');

const naturalLanguageUnderstanding = new NaturalLanguageUnderstandingV1({
  version: '2021-08-01',
  authenticator: new IamAuthenticator ({
    apikey: api_key
  }),
  serviceUrl: api_url
});
return naturalLanguageUnderstanding;
```

Part D: Create end points

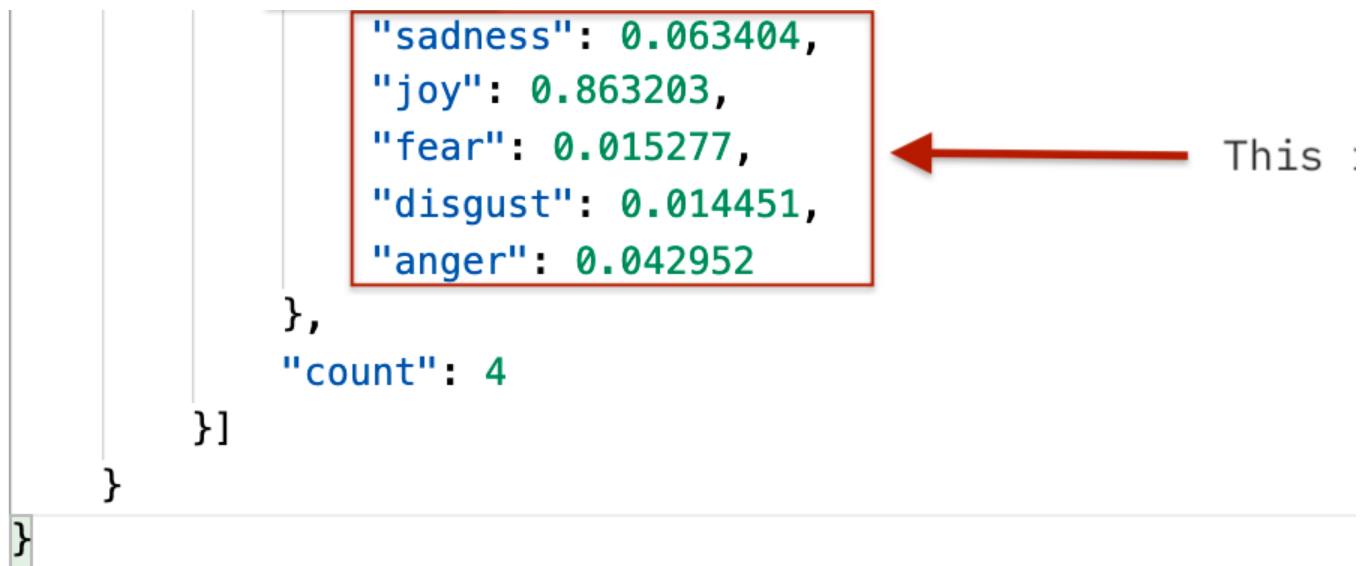
1. Observe that the server code intends to provide 4 end-points. One of the end-points has been provided for you. Uncomment it and observe the code. It is highly recommended that you attempt to implement these end-points by yourself in **sentimentAnalyzerServer.js** before you use the code given below to complete the assignment.

These are points you should keep in mind, while implementing the end-points.

- Each of the end points should use the `getNLUInstance()` created in the previous part to return appropriate response.
- Each end point should send a JSON or a string response as may be considered relevant.
- The JSON object returned from the IBM NaturalLanguageUnderstanding API is as in the below image. We need to extract the information that we need.

{

```
"status": 200, ← This says the call
"statusText": "OK",
"headers": {
  "server": "watson-gateway",
  "content-length": "496",
  "content-type": "application/json; charset=utf-8",
  "cache-control": "no-cache, no-store",
  "x-dp-watson-tran-id": "8ac7e28a-1490-4807-b6c7-b4d9",
  "content-security-policy": "default-src 'none'",
  "pragma": "no-cache",
  "x-content-type-options": "nosniff",
  "x-frame-options": "DENY",
  "x-xss-protection": "1; mode=block",
  "strict-transport-security": "max-age=31536000; incl",
  "x-request-id": "8ac7e28a-1490-4807-b6c7-b4d91f9ef9t",
  "x-global-transaction-id": "8ac7e28a-1490-4807-b6c7-",
  "x-edgeconnect-midmile-rtt": "276",
  "x-edgeconnect-origin-mex-latency": "2382",
  "date": "Mon, 23 Aug 2021 12:19:03 GMT",
  "connection": "close"
},
"result": {
  "usage": {
    "text_units": 2,
    "text_characters": 14211,
    "features": 1
  },
  "retrieved_url": "https://www.briantracy.com/blog/pe",
  "language": "en",
  "keywords": [{
    "text": "Inspirational quotes",
    "relevance": 0.727754,
    "emotion": {
```



1. /url/emotion end-point is used to analyze the **emotion** using NLU, for a given url.

```

app.get("/url/emotion", (req,res) => {
  let urlToAnalyze = req.query.url
  const analyzeParams =
  {
    "url": urlToAnalyze,
    "features": {
      "keywords": {
        "emotion": true,
        "limit": 1
      }
    }
  }

  const naturalLanguageUnderstanding = getNLUInstance();

  naturalLanguageUnderstanding.analyze(analyzeParams)
  .then(analysisResults => {
    //Retrieve the emotion and return it as a formatted string
    return res.send(analysisResults.result.keywords[0].emotion,null,2);
  })
  .catch(err => {
    return res.send("Could not do desired operation "+err);
  });
});

```

3. /url/sentiment end-point is used to analyze the **sentiment** using NLU, for a given url.

```

app.get("/url/sentiment", (req,res) => {
  let urlToAnalyze = req.query.url
  const analyzeParams =
  {
    "url": urlToAnalyze,
    "features": {
      "keywords": {
        "sentiment": true,
        "limit": 1
      }
    }
  }

  const naturalLanguageUnderstanding = getNLUInstance();

  naturalLanguageUnderstanding.analyze(analyzeParams)
  .then(analysisResults => {
    //Retrieve the sentiment and return it as a formatted string

    return res.send(analysisResults.result.keywords[0].sentiment,null,2);
  })
  .catch(err => {
    return res.send("Could not do desired operation "+err);
  });
});

```

4. /text/emotion end-point is used to analyze the **emotion** using NLU, for a given text.

```

app.get("/text/emotion", (req,res) => {
  let textToAnalyze = req.query.text
  const analyzeParams =
  {
    "text": textToAnalyze,
    "features": {
      "keywords": {
        "emotion": true,
        "limit": 1
      }
    }
  }

  const naturalLanguageUnderstanding = getNLUInstance();

  naturalLanguageUnderstanding.analyze(analyzeParams)
  .then(analysisResults => {
    //Retrieve the emotion and return it as a formatted string

    return res.send(analysisResults.result.keywords[0].emotion,null,2);
  })
  .catch(err => {
    return res.send("Could not do desired operation "+err);
  });
});

```

5. /text/sentiment end-point is used to analyze the **sentiment** using NLU, for a given text.

```

app.get("/text/sentiment", (req,res) => {
  let textToAnalyze = req.query.text
  const analyzeParams =
  {
    "text": textToAnalyze,
    "features": {
      "keywords": {
        "sentiment": true,
        "limit": 1
      }
    }
  }

  const naturalLanguageUnderstanding = getNLUInstance();

  naturalLanguageUnderstanding.analyze(analyzeParams)
  .then(analysisResults => {
    //Retrieve the sentiment and return it as a formatted string

    return res.send(analysisResults.result.keywords[0].sentiment,null,2);
  })
  .catch(err => {
    return res.send("Could not do desired operation "+err);
  });
});

```

Part E - React Client Side

1. Change to client directory sentimentAnalyzeClient.

```
cd /home/project/cazgi-IBM-Watson-NLU-Project/sentimentAnalyzeClient
```

{: codeblock}

2. All packages are needed to be installed are listed in package.json. To install and save those packages, run the following command in the terminal.

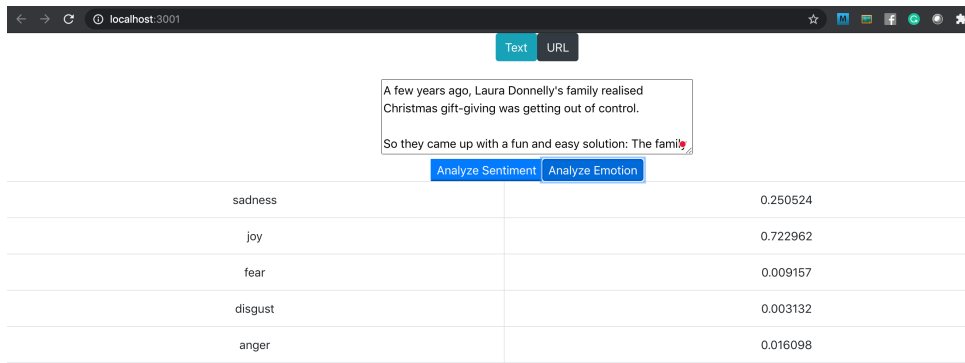
```
npm install -s
```

{: codeblock}

3. Make changes to client code

- Open the client code cazgi-IBM-Watson-NLU-Project/sentimentAnalyzeClient/public/index.html in the explorer
- Make changes to change the title of the page to "Sentiment Analyzer"
- Open the client code cazgi-IBM-Watson-NLU-Project/sentimentAnalyzeClient/src/App.js in the explorer
- Make changes to change the color of the sentiment response to green if the sentiment is positive, yellow if the sentiment is neutral and red if the sentiment is negative.

- Make changes in the `cazgi-IBM-Watson-NLU-Project/sentimentAnalyzeClient/src/EmotionTable.js` to use `map` function that we used in the Hands-on-ReactLab, to render the `EmotionTable` component as a below.



▼ [Click here to see how to implement Map](#)

```
Object.entries(this.props.emotions).map(function(mapentry) {
return (
  <tr>
    <td>{mapentry[0]}</td>
    <td>{mapentry[1]}</td>
  </tr>
)
})
```

4. Run `npm run-script build`. Please note this is customized in `package.json` to generate and copy the client code to the server.
5. Change to the server side.

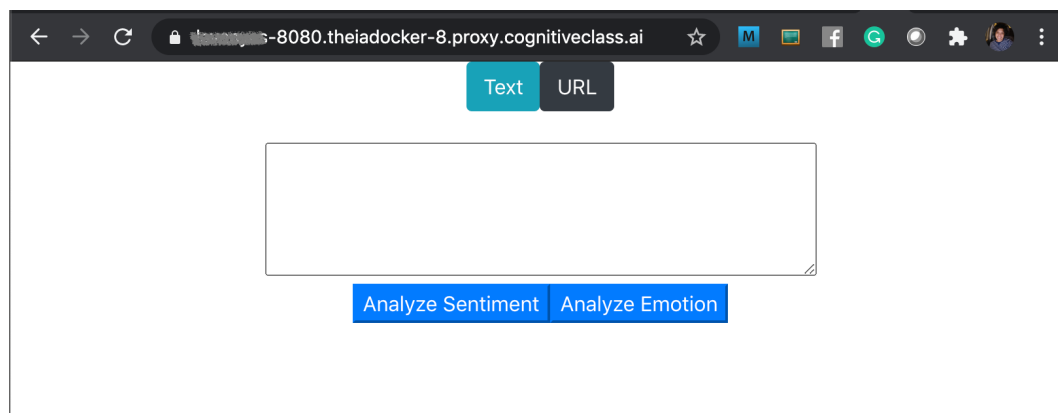
```
cd /home/project/cazgi-IBM-Watson-NLU-Project/sentimentAnalyzeServer
```

```
{: codeblock}
```

6. Run the server. This will start on port `8080`. `npm start`

```
{: codeblock}
```

7. Connect to port `8080` from **launch application**. Verify your output on the browser. The output depending on which whether you choose text or URL and whether you choose sentiment or emotions, the output should vary.



8. Please refer to this lab [Working with git in the Theia lab environment](#)

Part F: Commit and push your local code to your remote git repository

1. In the terminal, go to the project directory.

```
cd /home/project/cazgi-IBM-Watson-NLU-Project
```

```
{: codeblock}
```

2. Please refer to this lab [Working with git in the Theia lab environment](#)

- Run the following command to add all the files to the remote repository. *Remove the files that you have not changed from this list. Add any new files you have added to the list*

```
git add sentimentAnalyzeServer/package.json sentimentAnalyzeServer/package-lock.json sentimentAnalyzeServer/sentimentAnalyzerSer
```

```
{: codeblock}
```

- Run the following command to commit your changes. *Remove the files that you have not changed from this list. Add any new files you have added to the list*

```
git commit -m"Server and client side changes to implement the final project requirements" sentimentAnalyzeServer/package.json se
```

```
{: codeblock}
```

- Run the following command. You will be prompted by git for your username and password.

```
`git push`
```

- Type your GitHub username and enter the personal access token you generated, for password. When you are authenticated, all committed changes are synced with your GitHub repository.
- Share the URL of your Git repository with all the changes. *Ensure that the .env file with all the credentials is not added to the gitrepository.*

Checklist for completion

- Client-side changes done as per requirements.
 - Title changed to Sentiment Analyzer
 - Changed the color of the sentiments
 - Display response for emotion as a table
- Client side `npm run-script build` is run to build the client code and copy it to server side.
- .env file is created on the server side and the NLU credentials are added.
- All the four API endpoints have been implemented in the server side.
- Testing done from server side to see if the client is rendered as expected - Text Emotion, Text Sentiment, URL Emotions and URL Sentiment return results as expected.
- Code is pushed to Github.

Next step

The next and the final step is the deployment of the application on cloud foundry.

[Lavanya](#)

Other contributors

[Upkar Lidder](#)

Changelog

Date	Version	Changed by	Change Description
2022-06-29	0.5	K Sundararajan	Updated screenshots
2021-10-07	0.4	Sourabh	Updated Git push instructions
2021-01-20	0.3	Lavanya	Included npm version update
2021-01-20	0.2	Lavanya	Included git login commands
2020-12-01	0.1	Lavanya	Initial version created

© IBM Corporation 2020. All rights reserved.