Extract PDF Text and Generate Embeddings Using Hugging Face

Estimated time needed: 15 minutes

What you will learn

You will learn how to read PDF files in Node.js using the pdf-parse library to extract text content from the PDF. During the extraction process you will clean the text by removing newline characters and merging the text into a single continuous paragraph.

You will interact with external APIs, specifically Hugging Face's machine learning API, to convert extracted text into embeddings. This process involves sending text to the Hugging Face inference client, which then returns an array of embeddings generated by a pre-trained model, `sentence-transformers/all-MiniLM-L6-v2`.

Learning objectives

After completing this lab, you will be able to:

- Utilize the pdf-parse library to read a PDF file and extract its text content.
- Use the Hugging Face API to convert the extracted text into vector embeddings.
- Establish and utilize an inference client from Hugging Face to perform feature extraction.
- Implement asynchronous JavaScript using async/await to ensure appropriate ordering of the I/O operations such as reading PDF file.

Prerequisites

- Intermediate competency with JavaScript Node.js

Important notice about this lab environment

Skills Network Cloud IDE (based on Theia and Docker) is an open-source Integrated Development Environment (IDE) that provides an environment for hands-on labs in course and project-related labs.

Please be aware that sessions for this lab environment are not persistent. Every time you connect to this lab, a new environment is created for you.

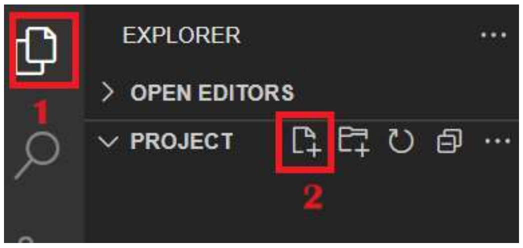
You will lose data if you exit the environment without saving to GitHub or another external source.

Plan to complete these labs in a single session to avoid losing your data.

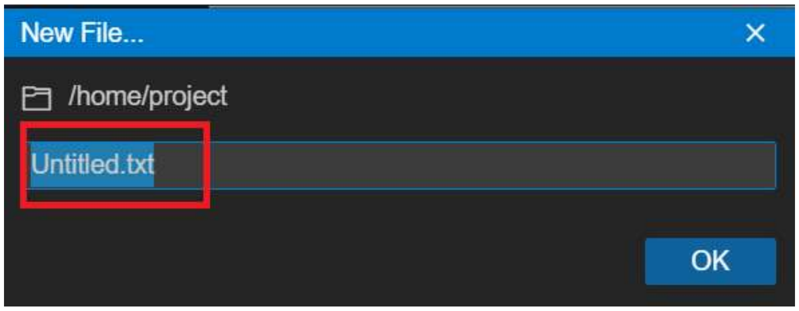
Extract text from PDF and convert to embeddings

Task 1: Create the JavaScript File

- Select **Explorer** on the left side of the terminal window, as shown at number 1 in the following screenshot. Then, within the **Project** folder, select **New File** as shown at number 2 in the following screenshot.



4. A pop-up box should display with the default file name **Untitled.txt**. Change the default name to `extract_pdf.js`.



Task 2: Install necessary packages

1. You need to install `pdf-parse` and `@huggingface/inference`.

- fs for file system operations (built-in with Node.js).
- pdf-parse to extract text from PDF files.
- @huggingface/inference to interact with the Hugging Face API for machine learning tasks.

```
npm install pdf-parse @huggingface/inference
```

2. Execute the following command to get PDF test file from which you will extract the text. This PDF contains a **Food Menu**.

```
wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/6z4EmYqfgIxyYLGukHedvA/foodMenu.pdf
```

Task 3: Import and Require

Import the necessary modules at the top of your file and initialize the Hugging Face inference client with your API key you obtained from the **Set Up Hugging Face API** lab.

```
const fs = require("fs");
const pdf = require("pdf-parse");
const { HfInference } = require("@huggingface/inference");
const hf = new HfInference("your_huggingface_api_key");
```

Note: The free version of the Hugging Face API key only allows a limited number of accesses in a given time period, although the documentation doesn't make that limit clear. It does seem like the limit is on an hourly basis, so if you get an error after frequent accesses to their API, you may need to wait up to an hour until you can continue your work.

Task 4: Extract text from a PDF file

You need to create to extract text from the PDF file and format it for further processing.

```
const extractTextFromPDF = async (filePath) => {
  try {
    const dataBuffer = fs.readFileSync(filePath);
    const data = await pdf(dataBuffer);
    const text = data.text.replace(/\n/g, " ").replace(/ +/g, " ");
    return text;
  } catch (err) {
    console.error("Error extracting text from PDF:", err);
    throw err;
  }
};
```

Let's review how this funciton works.

- The variable `const extractTextFromPDF` calls the anonomys asynchronous function `async (filePath) => { ... }` which takes a single parameter `filePath`. The string parameter represents the path to the PDF file from which you will extract text.
- A try-catch blocks wraps the entire function to handle any errors that might occur during the execution of the code inside the try block.
- The `dataBuffer` variable stores the results from the Node.js File System, `fs` module's `readFileSync()` method. It reads all content of the file specified by the `filePath` parameter.
- The `data` variable stores the return value from the anonomous asynchronous `pdf()` function from the `pdf-parse` library. It parses the PDF content stored in the `dataBuffer`. The `pdf()` function returns a promise that resolves to an object containing various details about the PDF, including the extracted text as a property called text.
- Next, you use the code `const text = data.text.replace(/\n/g, " ").replace(/ +/g, " ");` to format the extracted text.
  - The function `.replace(/\n/g, " ")` replaces all newline characters (\n) with a single space. It converts multi-line text into a single line or paragraph, making it easier to process further.
  - The function `.replace(/ +/g, " ")` replaces multiple consecutive spaces with a single space. It cleans irregular spacing in the text, resulting from the previous replacement or the original PDF formatting.
  - After formatting, `return text` returns the cleaned text.

Task 5: Convert the extracted text to an embedding

Now, you need to create one more function named `convertTextToEmbedding()` to generate embeddings from the extracted text.

```
const convertTextToEmbedding = async (text) => {
  try {
    const result = await hf.featureExtraction({
      model: "sentence-transformers/all-MiniLM-L6-v2",
      inputs: text,
    });
    // console.log("Embedding Result:", result);
    return result; // Return the embedding array
  } catch (err) {
    console.error("Error converting text to embeddings:", err);
    throw err;
  }
 };
```

- The asynchronous `convertTextToEmbedding()` function converts text into embeddings using the Hugging Face API.
- It first tries to call the `featureExtraction()` method with the specified model `sentence-transformers/all-MiniLM-L6-v2` and input text.

- If successful, it returns the resulting embeddings.
- If an error occurs, it logs the error and rethrows it. You can use the optional commented `console.log` line for debugging purposes to log the embedding result.

Now you should create a variable named `filePath` and initialize it with name of the PDF.

```
const filePath = "foodMenu.pdf";
```

Task 6: Create the main function

Now you need to create the `main()` function to invoke `convertTextToEmbedding()` and `extractTextFromPDF()`.

```
async function main(){
  const text = await extractTextFromPDF(filePath);
  console.log("Extracted Text:", text);
  const embeddings=await convertTextToEmbedding(text);
  console.log(embeddings);
}
```

The main function utilizes typical asynchronous JavaScript operations used for processing PDF text extraction and conversion to embeddings.

- By using async and await, it ensures that each step completes in order, handling the asynchronous nature of file I/O and processing operations gracefully.
- You will find this approach useful in scenarios where operations depend on the completion of previous tasks before proceeding, such as in data preprocessing or document analysis tasks.

Lastly, call the `main()` function in the last line of the `extract_pdf.js` file to kick-off the process.

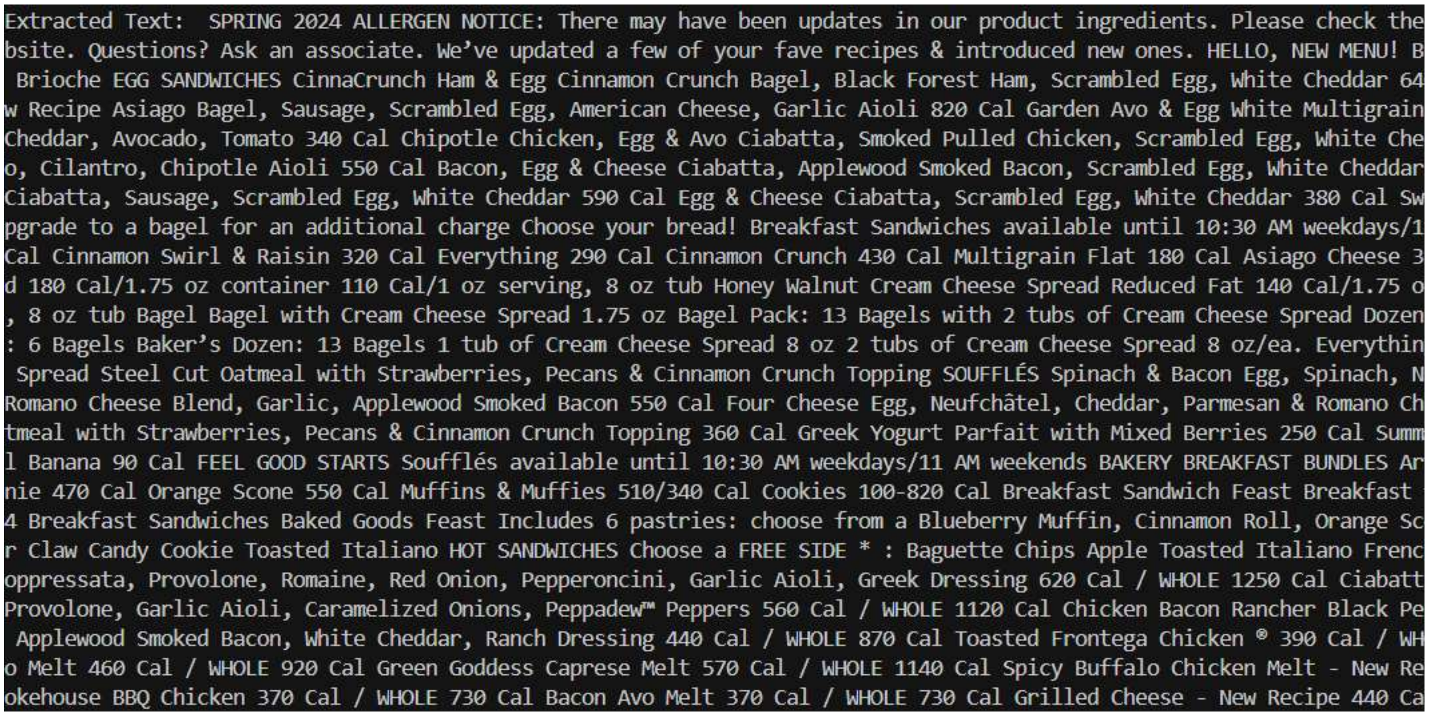
```
main()
```

Task 7: Check the output

1. Now open the terminal and perform given command to see the output.

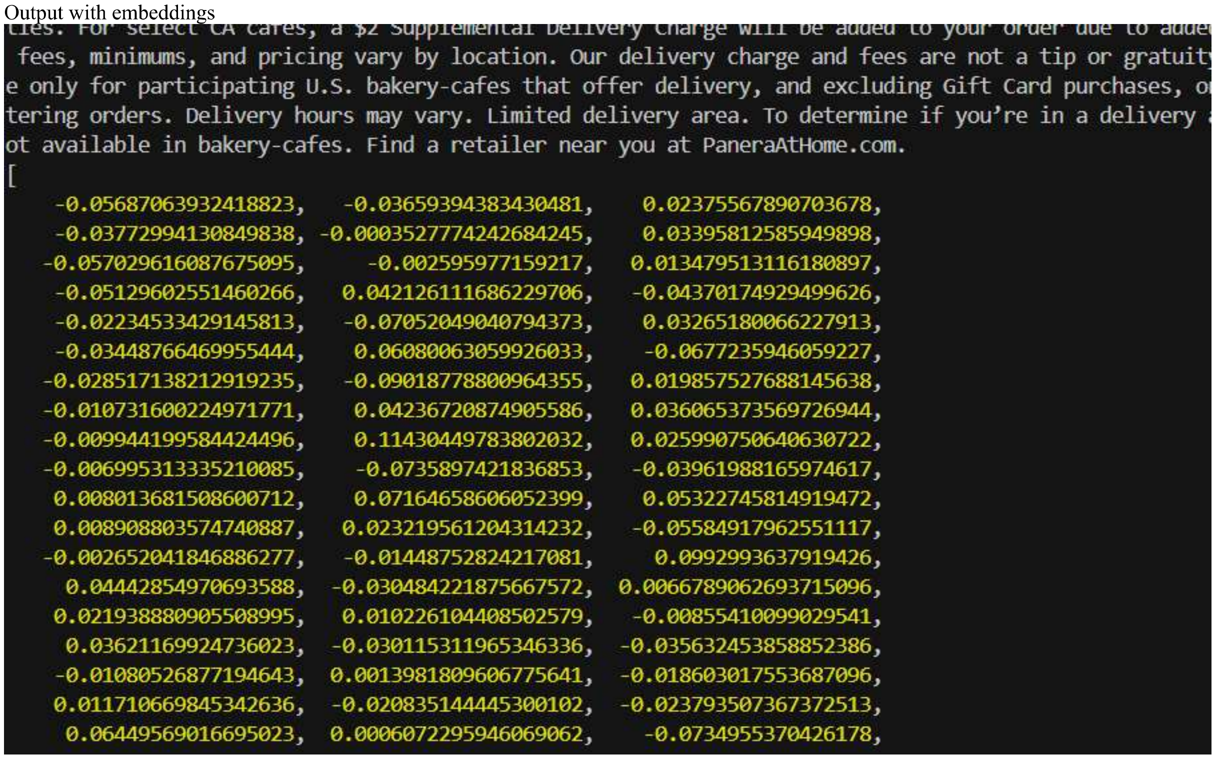
```
node extract_pdf
```

2. You will get the output text extracted from pdf as per given screenshot.



```
Extracted Text:  SPRING 2024 ALLERGEN NOTICE: There may have been updates in our product ingredients. Please check the
bsite. Questions? Ask an associate. We’ve updated a few of your fave recipes & introduced new ones. HELLO, NEW MENU! B
 Brioche EGG SANDWICHES CinnaCrunch Ham & Egg Cinnamon Crunch Bagel, Black Forest Ham, Scrambled Egg, White Cheddar 64
w Recipe Asiago Bagel, Sausage, Scrambled Egg, American Cheese, Garlic Aioli 820 Cal Garden Avo & Egg White Multigrain
Cheddar, Avocado, Tomato 340 Cal Chipotle Chicken, Egg & Avo Ciabatta, Smoked Pulled Chicken, Scrambled Egg, White Che
o, Cilantro, Chipotle Aioli 550 Cal Bacon, Egg & Cheese Ciabatta, Applewood Smoked Bacon, Scrambled Egg, White Cheddar
Ciabatta, Sausage, Scrambled Egg, White Cheddar 590 Cal Egg & Cheese Ciabatta, Scrambled Egg, White Cheddar 380 Cal Sw
pgrade to a bagel for an additional charge Choose your bread! Breakfast Sandwiches available until 10:30 AM weekdays/1
Cal Cinnamon Swirl & Raisin 320 Cal Everything 290 Cal Cinnamon Crunch 430 Cal Multigrain Flat 180 Cal Asiago Cheese 3
d 180 Cal/1.75 oz container 110 Cal/1 oz serving, 8 oz tub Honey Walnut Cream Cheese Spread Reduced Fat 140 Cal/1.75 o
, 8 oz tub Bagel Bagel with Cream Cheese Spread 1.75 oz Bagel Pack: 13 Bagels with 2 tubs of Cream Cheese Spread Dozen
: 6 Bagels Baker’s Dozen: 13 Bagels 1 tub of Cream Cheese Spread 8 oz 2 tubs of Cream Cheese Spread 8 oz/ea. Everythin
 Spread Steel Cut Oatmeal with Strawberries, Pecans & Cinnamon Crunch Topping SOUFFLÉS Spinach & Bacon Egg, Spinach, N
Romano Cheese Blend, Garlic, Applewood Smoked Bacon 550 Cal Four Cheese Egg, Neufchâtel, Cheddar, Parmesan & Romano Ch
tmeal with Strawberries, Pecans & Cinnamon Crunch Topping 360 Cal Greek Yogurt Parfait with Mixed Berries 250 Cal Summ
l Banana 90 Cal FEEL GOOD STARTS Soufflés available until 10:30 AM weekdays/11 AM weekends BAKERY BREAKFAST BUNDLES Ar
nie 470 Cal Orange Scone 550 Cal Muffins & Muffies 510/340 Cal Cookies 100-820 Cal Breakfast Sandwich Feast Breakfast
4 Breakfast Sandwiches Baked Goods Feast Includes 6 pastries: choose from a Blueberry Muffin, Cinnamon Roll, Orange Sc
r Claw Candy Cookie Toasted Italiano HOT SANDWICHES Choose a FREE SIDE * : Baguette Chips Apple Toasted Italiano Frenc
oppressata, Provolone, Romaine, Red Onion, Pepperoncini, Garlic Aioli, Greek Dressing 620 Cal / WHOLE 1250 Cal Ciabatt
Provolone, Garlic Aioli, Caramelized Onions, Peppadew™ Peppers 560 Cal / WHOLE 1120 Cal Chicken Bacon Rancher Black Pe
 Applewood Smoked Bacon, White Cheddar, Ranch Dressing 440 Cal / WHOLE 870 Cal Toasted Frontega Chicken ® 390 Cal / WH
o Melt 460 Cal / WHOLE 920 Cal Green Goddess Caprese Melt 570 Cal / WHOLE 1140 Cal Spicy Buffalo Chicken Melt - New Re
okehouse BBQ Chicken 370 Cal / WHOLE 730 Cal Bacon Avo Melt 370 Cal / WHOLE 730 Cal Grilled Cheese - New Recipe 440 Ca
```

- Output with embeddings



```
ties. For select CA cafes, a $2 Supplemental Delivery Charge will be added to your order due to adde
 fees, minimums, and pricing vary by location. Our delivery charge and fees are not a tip or gratuit
e only for participating U.S. bakery-cafes that offer delivery, and excluding Gift Card purchases, o
tering orders. Delivery hours may vary. Limited delivery area. To determine if you’re in a delivery
ot available in bakery-cafes. Find a retailer near you at PaneraAtHome.com.
[
    -0.05687063932418823,   -0.03659394383430481,    0.02375567890703678,
    -0.03772994130849838, -0.0003527774242684245,    0.03395812585949898,
   -0.057029616087675095,     -0.002595977159217,   0.013479513116180897,
    -0.05129602551460266,   0.042126111686229706,   -0.04370174929499626,
    -0.02234533429145813,   -0.07052049040794373,    0.03265180066227913,
    -0.03448766469955444,    0.06080063059926033,    -0.0677235946059227,
   -0.028517138212919235,   -0.09018778800964355,   0.019857527688145638,
   -0.010731600224971771,    0.04236720874905586,   0.036065373569726944,
   -0.009944199584424496,    0.11430449783802032,   0.025990750640630722,
   -0.006995313335210085,    -0.0735897421836853,   -0.03961988165974617,
    0.008013681508600712,    0.07164658606052399,    0.05322745814919472,
    0.008908803574740887,   0.023219561204314232,   -0.05584917962551117,
   -0.002652041846886277,   -0.01448752824217081,     0.0992993637919426,
     0.04442854970693588,  -0.030484221875667572,  0.0066789062693715096,
    0.021938880905508995,   0.010226104408502579,   -0.00855410099029541,
     0.03621169924736023,  -0.030115311965346336,  -0.035632453858852386,
    -0.01080526877194643,  0.0013981809606775641,  -0.018603017553687096,
    0.011710669845342636,  -0.020835144445300102,  -0.023793507367372513,
     0.06449569016695023,  0.0006072295946069062,    -0.0734955370426178,
```

Conclusion

- The provided code utilizes Node.js and libraries such as `fs` for file system operations, `pdf-parse` for PDF parsing, and the Hugging Face `@huggingface/inference` library for text embeddings.
- The `extractTextFromPDF()` function reads a PDF file synchronously using `fs.readFileSync`, parses it using `pdf-parse`, and cleans up the extracted text by replacing newline characters and multiple spaces.
- The `convertTextToEmbedding()` function utilizes the Hugging Face model `sentence-transformers/all-MiniLM-L6-v2` to convert the extracted text into embeddings, which represent a numerical representation of the text's meaning.
- The `main()` function sequentially calls `extractTextFromPDF()` and `convertTextToEmbedding()`, logging the extracted text and embeddings to the console.

Author(s)

Richa Arora