

Project Documentation

Scripts Overview

Entity Data Extraction

1. content_extract.ipynb

Purpose

This script aims to extract structured data from PDF documents using Google Cloud Document AI. It utilizes optical character recognition (OCR) to identify entities like names, ranks, and regiments from the provided document.

Dependencies

- **google-cloud-documentai:** For interfacing with Google Cloud Document AI.
- **google-cloud-storage:** For storing documents.
- **aspose-words:** For breaking pdfs into pages.

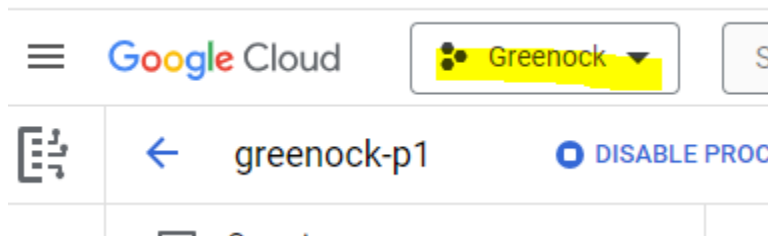
Functionality

1. **Data Extraction:** Extracts structured data such as last name, first name, rank, regiment, and page number from the provided PDF document.
2. **CSV Output:** Saves the extracted data into a CSV file.

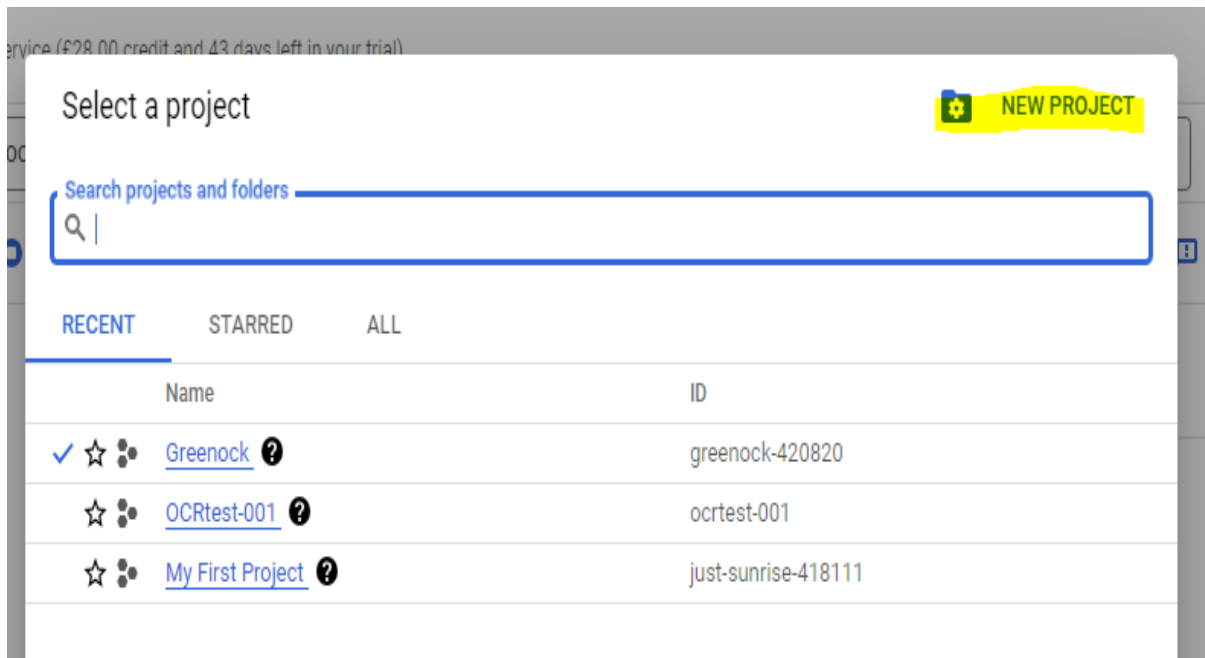
Google Cloud Document AI Integration:

Create Project

1. Go to Google Cloud Console
2. Click [here](#)



3. Click on New Project



4. Give the project a name

☰ Google Cloud

New Project

⚠ You have 22 projects remaining in your quota. Request an increase or delete projects. [Learn more](#)

[MANAGE QUOTAS](#)

❗ Make sure all fields are correct to continue

Project name * ?

Project ID: french-prisoners. It cannot be changed later. [EDIT](#)

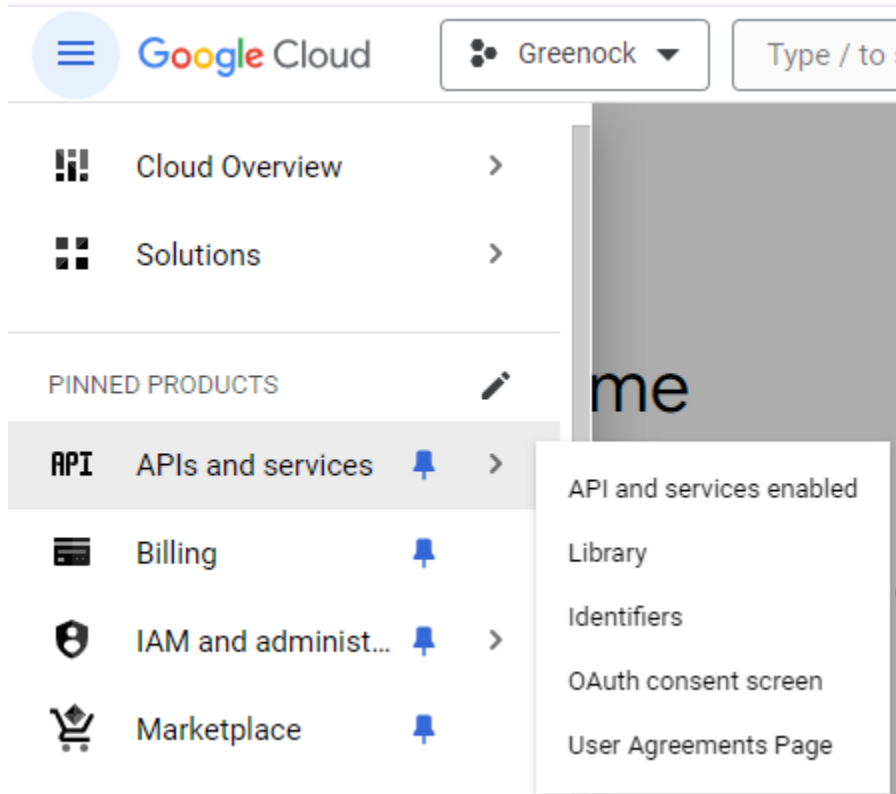
Location * [BROWSE](#)

Parent organization or folder

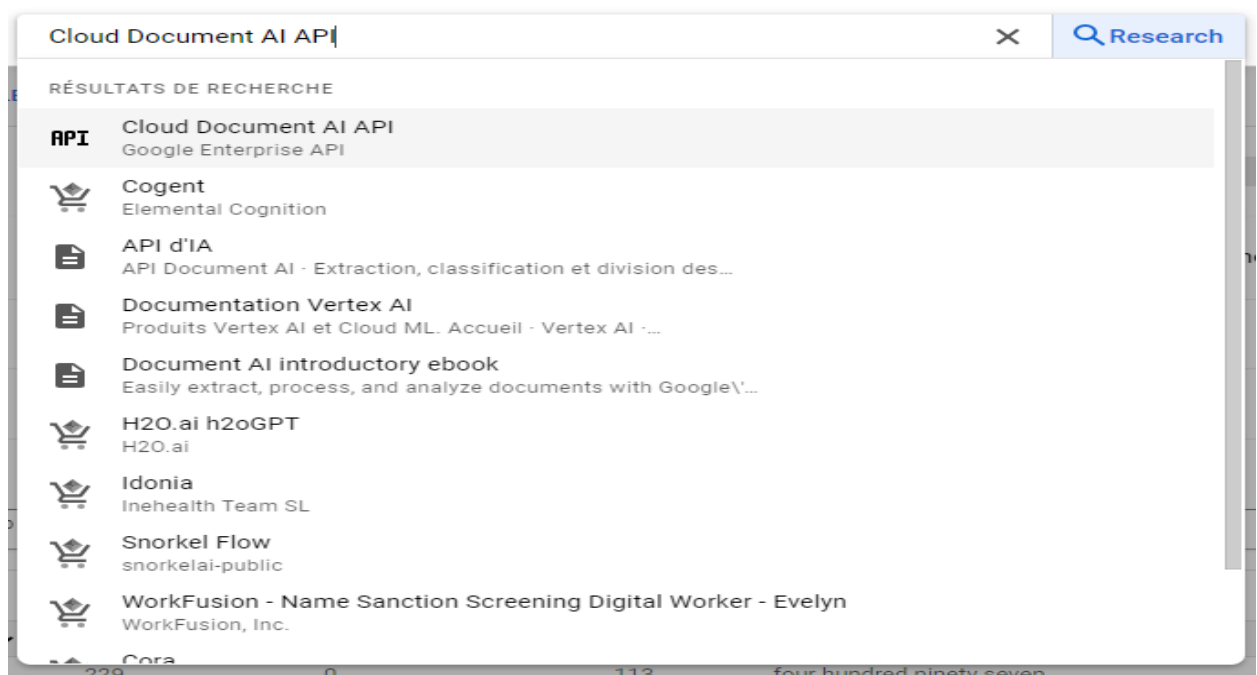
[CREATE](#) [CANCEL](#)

Cloud Document AI API

1. Go to the "API & Services" section in the Google Cloud Console.




2. Click on "Enable APIs and Services" and search for "Cloud Document AI API".



3. Select the Document AI API from the search results and click "Enable".

[←](#) Product details

[Back to Marketplace](#)



Cloud Document AI API

[Google Enterprise API](#)

Service to parse structured information from unstructured or semi-structured documents using...

[ENABLE](#) [TRY THIS API ↗](#)

OVERVIEW

PRICING

DOCUMENTATION

RELATED PRODUCTS

Document AI

1. After enabling the API, search "Document AI" and then you'll be redirected to the Document AI dashboard.
2. This screen will appear

Preview [LEARN](#)

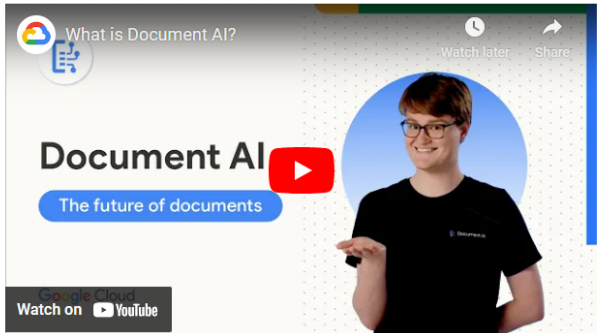
Getting started with Document AI

Document AI lets you transform dark, unstructured documents into actionable data to drive operational efficiencies, simplify business processes, and make better decisions.

[EXPLORE PROCESSORS](#) [CREATE A CUSTOM PROCESSOR](#)

Check out our tutorials to learn how to train, benchmark, and deploy a custom Document AI processor or Workbench processor.

[VIEW TUTORIALS](#)



What is Document AI?

Document AI

The future of documents

Watch on YouTube

3. Click on "Create a Custom Processor"

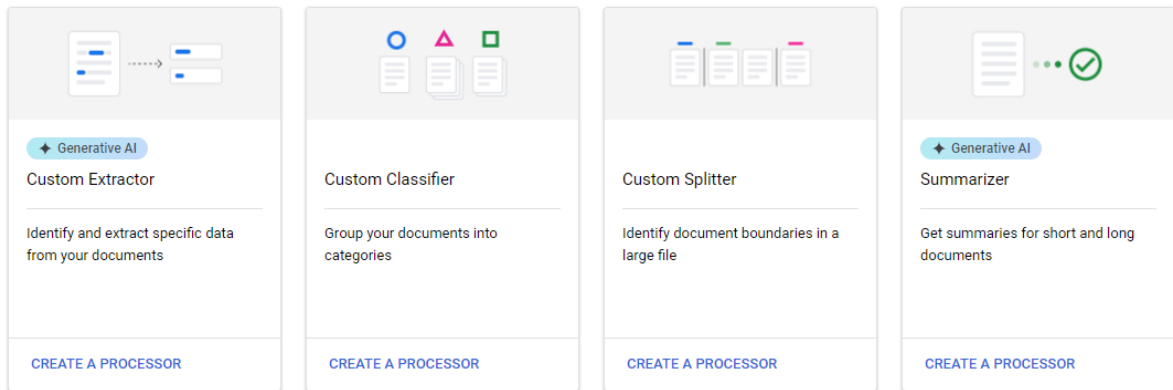
[EXPLORE PROCESSORS](#) [CREATE A CUSTOM PROCESSOR](#)

4. This screen will appear

Workbench

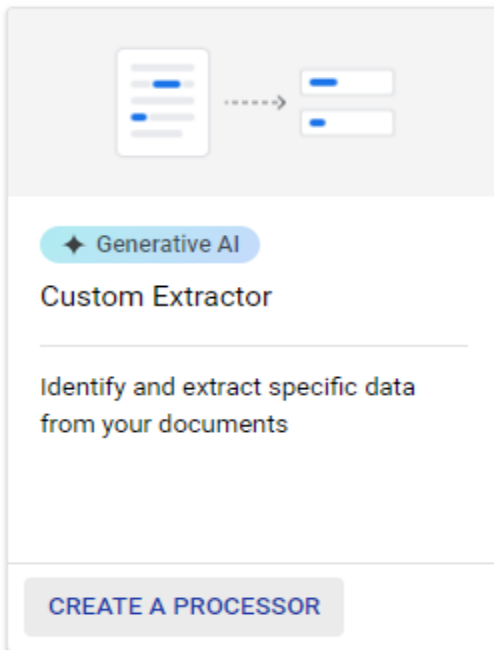
Processors for Document AI Workbench allow you to quickly generate predictions with generative AI or train your own custom processors from scratch.

[DOCAI WORKBENCH REVIEWS](#)



Creating Processor

1. Under Custom Extractor click on “Create a Processor”



2. Give your processor name and then click on “Create”

Custom Extractor

Train a custom ML model to identify and extract custom entities, checkboxes, and other form elements from documents using just 10 examples [Learn more](#)

Processing tool name *

example

Must start with a letter. May contain letters, numbers, spaces, hyphens, and underscores.

Region

US (United States)

ADVANCED OPTIONS

CREATE

CANCEL

3. After creating this screen will appear.

Preview
Processing tool details

First steps
Set schema and preview

Build
Create versions

Evaluate and test
Version performance

Deploy and use
Manage versions

Preview

Use your processor

Ready to use
Call your processor's out-of-the-box base model to get immediate predictions
[GET A SAMPLE API REQUEST](#)

Personalize
Improves the accuracy of predictions for your documents
[First steps](#)

4. Then click on “First Steps”

Preview
Processing tool details

First steps
Set schema and preview

Build
Create versions

Evaluate and test
Version performance

Deploy and use
Manage versions

First steps
Identify and extract specific data from your documents

Fields [+ CREATE A FIELD](#) [IMPORT A SAMPLE DOCUMENT](#)

Filtered

 Saisissez le nom ou la valeur de la propriété

	Name	Data type	Occurrence	Enabled
No results to display				

Schema Definition

Began by defining the schema of the documents in Document AI. This involved specifying the key information we wanted to extract, such as names, ranks, and regiments.

1. Click on “Create A Field” and define Label Name, Data type, and Occurrence.

Create a label

Name *
Label Name

Must start with a letter. May contain letters, numbers, spaces, hyphens, and underscores.

☐ This is a parent label **BETA** ?

Data type
Plain text

Occurrence
Optional multiple

[CANCEL](#) [CREATE](#)

2. Define all the labels you want to extract from your data.

Get started

Identify and extract specific data from your documents

Fields

[+ CREATE NEW FIELD](#) [↑ UPLOAD SAMPLE DOCUMENT](#)

Filter Enter property name or value

<input type="checkbox"/>	Name ↑	Data type	Occurrence	Enabled	
<input type="checkbox"/>	first_name	Plain text	Optional multiple	✓ Yes	⋮
<input type="checkbox"/>	last_name	Plain text	Optional multiple	✓ Yes	⋮
<input type="checkbox"/>	page	Plain text	Optional multiple	✓ Yes	⋮
<input type="checkbox"/>	rank	Plain text	Optional multiple	✓ Yes	⋮
<input type="checkbox"/>	regiment	Plain text	Optional multiple	✓ Yes	⋮

3. After creating fields click on “Upload Sample Document”

Apply labels to documents to define your fields and improve processor quality.

Accepted formats: JPEG, JPG, PNG, BMP, PDF, TIFF, TIF, GIF [Learn more](#)

Select an import method

- ☒ Import documents from your local computer
- ☐ Import documents from Google Cloud Storage

BROWSE

IMPORT

CANCEL

4. Now import a sample document from where ever you want, and then click “Import”

Select an import method

- ☒ Import documents from your local computer
- ☐ Import documents from Google Cloud Storage

00000018.jpg

✕ BROWSE

IMPORT

CANCEL

5. Label your document according to fields.

Filtered Enter text to filter results

+ CREATE A FIELD

CLEAR SUGGESTIONS

first_name JAMES

first_name ADAM

first_name ADAM,

first_name ANDREW W.

first_name DAVID

first_name JOSEPH

first_name SIMON

first_name WILLIAM

MARK AS TAGGED

NAME.	UNIT.
ABBOTT, JAMES, Sergeant	6th Royal Scots Fusiliers
ADAMS, ADAM, Private	and Highland Light Infantry
ADAMS, ANDREW W., Driver	Royal Army Service Corps
ADAMS, DAVID, Private	13th Royal Scots
ADAMS, JOSEPH, Private	20th King's London Regiment
ADAMS, SIMON, Sapper	Royal Engineers
ADAMS, WILLIAM, Private	11th Argyll and Sutherland Highlanders
ADAMSON, JAMES, Private	6th Connaught Rangers
AINSIE, JOHN E., and Lieutenant	6th Royal Scots
ARMEN, ANDREW, and Lieutenant	South African Expeditionary Force
ARMEN, CHARLES, Private	6th Argyll and Sutherland Highlanders
ARMEN, JAMES, Sergeant	Royal Engineers
ARMEN, ROBERT, Private	and Gordon Highlanders
ARMEN, WILLIAM, Private	16th Royal Scots
ALEXANDER, DONALD, Private	11th Highland Light Infantry
ALEXANDER, EDWARD, Private	8th Argyll and Sutherland Highlanders
ALEXANDER, EDWARD, Lance-Corporal	7th King's Own Scottish Borderers
ALEXANDER, HUMPHREY, Driver	Royal Army Service Corps
ALEXANDER, WILLIAM, Chief Engineer	Transport Service
ALLAN, GEORGE W., Private	1st Highland Light Infantry
ALLAN, GORDON S., Private	8th Seaforth Highlanders
ALLAN, JOHN, Lance-Corporal	2nd Seaforth Highlanders
ALLAN, JOHN B., Corporal	and Argyll and Sutherland Highlanders
ALLAN, PETER R., Private	South African Expeditionary Force
ALLAN, FRANK D., Private	4th Argyll and Sutherland Highlanders
ALLAN, THOMAS H., Lance-Corporal	4th Argyll and Sutherland Highlanders
ALLISON, RICH, Private	2nd Seaforth Highlanders
ALLISON, SAMUEL, Gunner	Royal Field Artillery
ANDERSON, ARCHIBALD McE.	Transport Service
ANDERSON, GEORGE A., Lieutenant	6th Black Watch
ANDERSON, GRAEME S., Corporal	8th Seaforth Highlanders

Data Preparation

Next, gathered a dataset of sample documents that were representative of the documents intended to process. These documents were manually labeled to identify the target entities and their locations within the documents.

1. Click on “Build”. This screen will appear

Preview
Processing tool details

First steps
Set schema and preview

Build
Create versions

Evaluate and test
Version performance

Deploy and use
Manage versions

Build
Create processor builds for testing and deployment

NEXT STEP: EVALUATE AND TEST

Overview of the dataset

START LABELING IMPORT DOCUMENTS MANAGE DATASET

Total number of documents	With label	Without label	Automatic label	Suggestions
2	0	2	0	0
2 training, 0 tests, 0 unassigned	0 training, 0 tests, 0 unassigned	2 training, 0 tests, 0 unassigned	0 training, 0 tests, 0 unassigned	0 training, 0 tests, 0 unassigned

Create versions

You don't know which version to compile? Check out our [version guide](#) to understand each method and our [pricing guide](#) for cost information.

Call a foundation model
Create a training-free version using a Google foundation template and the fields you created

Setting
Tune a foundation model effortlessly using a labeled data set and fields you create

2. Click on “Import Documents” and give path of your documents, then select Data Select Data distribution.

Select an import method

- ☒ Import documents from Google Cloud Storage
- ☐ Import documents from your local computer

Select a source folder in Cloud Storage

Enter the Cloud Storage paths to the folders where your documents are stored. Your documents will be imported recursively.

[Data distribution](#)

ADD ANOTHER FOLDER

Automatic labeling
Start adding labels to your documents using predictions from existing versions of your processor. [See an example](#)

Automatic distribution (recommended)
Assign documents 80% for practice and 20% for test

Training
Used to train the machine learning model

Test
Used to evaluate the machine learning model

Not attributed

3. Then check “Import with automatic labeling”, and if you are importing document first time you can select pretrained version for automatic labeling, and then click on import.

Automatic labeling

Start adding labels to your documents using predictions from existing versions of your processor. [See an example](#)

- ☒ Import with automatic labeling

Version *
pretrained-foundation-model-v1.1-2024-03-12

Only deployed versions can be used for automatic labeling.

IMPORT

CANCEL

4. Then you have to review auto labeled data.

5. Check if all the fields are correctly labeled, if not then label incorrect or missing fields manually, and after doing labeling click on “Mark as Tagged”. Label all the documents.

The screenshot shows a web application interface. On the left, there is a list of names with a search bar at the top containing "CLEAR SUGGESTIONS". The names are: first_name JAMES, first_name ADAM, first_name ADAM, first_name ANDREW W., first_name DAVID, first_name JOSEPH, first_name SIMON, and first_name WILLIAM. Each name is preceded by a purple square icon with a white plus sign. To the right of the list is a large image of a document titled "The Men of Greenock who fell in the Great War, 1914-1918." The document is a list of names and their service details, organized in two columns: "Name" and "Unit". The names are listed in two columns, and the units are listed in two columns. The names are: JAMES, ADAM, ADAM, ANDREW W., DAVID, JOSEPH, SIMON, and WILLIAM. The units are: 1st Bn. Greenock Rifles, 2nd Bn. Greenock Rifles, 3rd Bn. Greenock Rifles, 4th Bn. Greenock Rifles, 5th Bn. Greenock Rifles, 6th Bn. Greenock Rifles, 7th Bn. Greenock Rifles, and 8th Bn. Greenock Rifles.

6. All the documents should be labeled.

Overview of the dataset

START LABELING

IMPORT DOCUMENTS

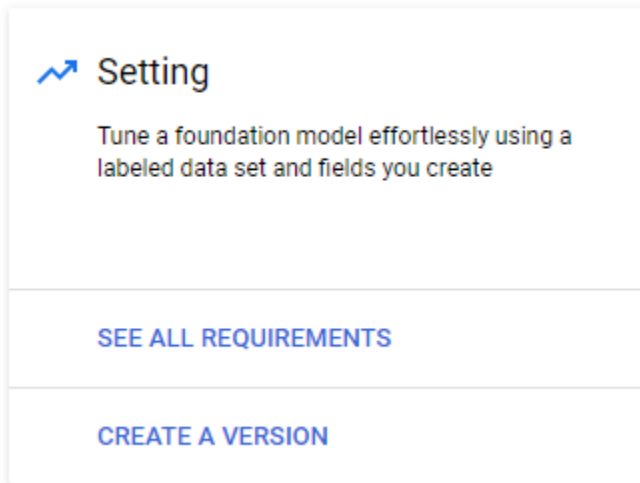
MANAGE DATASET

Total number of documents	With label ?	Without label	Automatic label	Suggestions ?
39	39	0	0	0
29 training, 10 tests, 0 unassigned	29 training, 10 tests, 0 unassigned	0 training, 0 tests, 0 unassigned	0 training, 0 tests, 0 unassigned	0 training, 0 tests, 0 unassigned

Training Version

With the labeled dataset prepared, initiated the fine-tuning process by creating version.

1. After labeling all the documents, we'll create a version.



2. Before creating version see all requirements. You can only perform fine-tuning if your labels are meeting minimum requirements.

Version requirements

Training instructions

- ⚠ Each label appears on 50 training documents (minimum: 10)
- ⚠ Each label appears on 50 test documents (minimum: 10)

⚠ Your dataset does not meet the recommended training criteria. You can still train a new version, but the quality of your model may be affected.

Tagged Documents

Labels	Documents	● Training	● Test
first_name	<div><div></div><div></div></div> 39	29	10
last_name	<div><div></div><div></div></div> 39	29	10
page	<div><div></div><div></div></div> 39	29	10
rank	<div><div></div><div></div></div> 39	29	10
regiment	<div><div></div><div></div></div> 39	29	10

CLOSE

- If you are meeting requirements, then create version by defining version name, and click “Create”.

Create a version

Set a foundation model

Tune a large model effortlessly using a labeled dataset and fields you create. [Learn more](#)
[about versioning methods.](#)

Version name *

Must start with a letter. May contain letters, numbers, spaces, hyphens, and underscores.

SHOW ADVANCED OPTIONS

CREATE CANCEL

- After creating version training will automatically gets start and you can view it in “Manage Dataset”

Manage dataset

All the documents 39
With label 39
Automatic label 0
Without label 0
Suggestions 0 NEW 0

IMPORT DOCUMENTS

Filtered Filtrer les éléments

Select all

Training

example-version Training...

TRAIN A NEW VERSION

Or you can also view it in “Deploy and use” page

greenock-p1 DISABLE CPU ACTIVITY DOCAI WORKBENCH REVIEWS

Preview Processing tool details
First steps Set schema and preview
Build Create versions
Evaluate and test Version performance
Deploy and use Manage versions

MANAGE VERSIONS

Your default version is managed by Google and automatically updated at regular intervals. You will be notified before each upgrade.

Default version pretrained-foundation-model-v1.0-2023-08-22 ABOUT GOOGLE UPGRADES VIEW THE RELEASE NOTES

The default version Allows you to process documents posted to your processor's prediction endpoint URL.

Releases DEPLOY CANCEL DEPLOYMENT COMPARE IMPORT

Filtered Filtrer les versions

Version ID	Creation	State	Name	Kind	F1 score	
e3b80ec68318ac24	May 13, 2024, 3:03:27 p.m.	Training...	example-version	Generative AI	--	SHOW

- After completion of training, select the version and then “Deploy” to deploy the version.

Releases						
DEPLOY CANCEL DEPLOYMENT COMPARE IMPORT						
Filtered Filtrer les versions ?						
	Version ID	Creation ↓	State	Name	Kind	F1 score ?
<input checked="" type="checkbox"/>	e3b80ec68318ac24	May 13, 2024, 3:03:27 p.m.	Training...	example-version	Generative AI	--

Evaluation and Validation

After training, done evaluation and validation tests to ensure the accuracy and effectiveness of the trained version. Tested the version and measured performance metrics such as F1 score.

- After deploying go to “Evaluate and test” page to evaluate version by selecting it from dropdown.

Preview
Processing tool details

First steps
Set schema and preview

Build
Create versions

Evaluate and test
Version performance

Deploy and use
Manage versions

Evaluate and test

Understand your release's performance before deployment and use

Version

pretrained-foundation-model-v1.0-2023-08-22

SEE THE FULL REVIEW

RUN A NEW ASSESSMENT

Presentation

Name	-
State	Unspecified
Kind	Generative AI
Creation	-
Latest review	-
Prediction endpoint	
F1 score ?	0.0
Precision ?	Not available
Reminder ?	Not available

- Test your version on document by clicking on “Import a Test Document”.


Test this version


Accepts JPEG, JPG, PNG, BMP, PDF, TIFF, TIF, GIF (15 pages, 20 MB maximum)

[IMPORT A TEST DOCUMENT](#)

- After evaluating and testing, if your version requires improvements then you have to expand your dataset and label more documents.

Improve your processor
























**Expand your dataset**
Go to the Create page to import additional training and testing documents for labeling
[→ Access the "Create" page](#)

**Tag more documents**
Labeling more instances of each field can improve the accuracy of a version
[→ Continue to label](#)

Processor Deployment


Once the version was trained and validated, deployed the Document AI processor to the Google Cloud platform. This allowed to utilize the processor programmatically to extract data from documents.

- If your version has good F1-score and performing well on test documents. Then proceed to "Deploy and use" page again.


 Preview Processing tool details	MANAGE VERSIONS					
 First steps Set schema and preview	Filtered Filtrer les versions ? <input checked="" type="checkbox"/> Partial match ?					
 Build Create versions	State	Name	Kind	F1 score ?	API	
 Evaluate and test Version performance		example-version		--	SHOW THE DETAILS	EXAMPLE QUERY 
 Deploy and use Manage versions		greenock-v2		0.979	SHOW THE DETAILS	EXAMPLE QUERY 
		greenock-v1		0.983	SHOW THE DETAILS	EXAMPLE QUERY 
		greenock-v1		N / A	SHOW THE DETAILS	EXAMPLE QUERY 
		Google Release Candidate		COMPLETE THE ASSESSMENT FIRST	SHOW THE DETAILS	EXAMPLE QUERY 
		Google Stable		0.98	SHOW THE DETAILS	EXAMPLE QUERY 

- Click on "Example Query" of your selected version.

API




[EXAMPLE QUERY](#) 

3. Click on "Python".

Detailed documentation on API requests is available here: [Submit a processing request](#) 

REST **PYTHON**

You can use a simple Python script to retrieve predictions from your active endpoint.

1. Make sure you have installed the [Google Cloud SDK](#) .
2. Follow the setup instructions described at [Process documents using client libraries](#) .
3. Copy and paste the sample code available in GitHub to a new Python file:
https://github.com/GoogleCloudPlatform/python-docs-samples/blob/main/documentai/snippets/process_document_sample.py 
4. Run your query in Python.

```
process_document_sample (  
    project_id = "322996871040" ,  
    location= "us" ,  
    processor_id= "eb7f025d9a48d15" ,  
    file_path= "/path/to/local/pdf" ,  
    mime_type= "application/pdf" ,  
)
```



4. You can copy processor version from here.

Filtered Filtrer les versions ?						
<input type="checkbox"/>	Version ID	Creation ↓	State	Name	Kind	F1 score ?
<input type="checkbox"/>	e3b80ec68318ac24	May 13, 2024, 3:03:27 p.m.	Training...	example- version	Generative AI	--

Integration with Content Extraction Script

In the content_extract.ipynb script, the Document AI processor is utilized to perform data extraction tasks. Here's how it integrates into the workflow:

1. **Initialization:** The script initializes the Document AI client and specifies the processor to use for document processing.
2. **Document Processing:** For each page of the PDF document, the script sends the image content to the Document AI processor for analysis. The processor applies OCR and extracts structured data based on the defined entity types.
3. **Data Extraction:** The extracted data, including entities like names, ranks, and regiments, is then processed further to format it appropriately and save it into a CSV file for easy analysis.

Usage

1. Open the content_extract.ipynb notebook in your Jupyter environment.
2. Provide the necessary input parameters such as project ID, location, processor ID, path (directory or file path), and processor version ID.
3. Execute the script to extract data from the PDF document.
4. Review the extracted data saved in the output CSV file.

```
determine_process(  
    project_id="322996871040",  
    location="us",  
    processor_id="eb7f025d9a48d15",  
    path="test",  
    #processor_version_id="69f0b6febc1d5673"  
)
```

The screenshot shows a code snippet where the default processor version is used in the `determine_process` function. This decision is based on testing, where it was found that the default version provided better results in terms of accuracy and efficiency compared to other versions.

Word Search

2. coordinates.ipynb

Purpose

This script is designed to locate and visualize the coordinates of specific words within a PDF document. It utilizes the Google Vision API for optical character recognition (OCR) and then identifies the coordinates of the specified words.

Dependencies

- **PyMuPDF:** For working with PDF documents.
- **Pillow:** For image processing tasks.
- **requests:** For making HTTP requests.

Functionality

1. **OCR and Data Extraction:** Extracts text and coordinates from the PDF document using the Google Vision API.
2. **Word Coordinate Search:** Searches for the specified word within the extracted data and retrieves its coordinates.
3. **Visualization:** Draws bounding boxes around the specified word on each page of the PDF document and saves the modified pages as images.

Coordinate Extraction and Bounding Box Drawing:

Text and Coordinate Extraction:

API Generation: The script utilizes the Google Cloud Vision API to process the PDF document. To use the API, we need to generate an API key through the Google Cloud Platform (GCP) Console. This API key is then used to authenticate requests sent to the Vision API.

Credentials

[+ CREATE CREDENTIALS](#)
[DELETE](#)
[RESTORE DELETED CREDENTIALS](#)

Create credentials to access the API

Remember this

API key

Identifies your project using a simple API key to check quota and access

OAuth client ID

Requests user consent so your app can access the user's data

Service account

Enables server-to-server, app-level authentication using robot accounts

CONFIGURE CONSENT SCREEN

API Keys

☐ Name

Help me choose

☐ [API key](#)

Asks a few questions to help you decide which type of credential to use

Restrictions

None

Actions

SHOW KEY

API Usage: With the API key in hand, the script makes HTTP requests to the Vision API, passing the PDF document's image content. The API returns a JSON response containing the detected text and the bounding box coordinates of each word on every page of the document.

```
[{'text': 'The', 'vertices': [{'x': 106, 'y': 163}, {'x': 145, 'y': 163}, {'x': 145, 'y': 181}, {'x': 106, 'y': 181}], 'page': 1}, {'text': 'Men', 'vertices': [{'x': 154, 'y': 163}, {'x': 201, 'y': 163}, {'x': 201, 'y': 181}, {'x': 154, 'y': 181}], 'page': 1}, {'text': 'of', 'vertices': [{'x': 212, 'y': 163}, {'x': 230, 'y': 163}, {'x': 230, 'y': 181}, {'x': 212, 'y': 181}], 'page': 1}, {'text': 'Greenock', 'vertices': [{'x': 239, 'y': 163}, {'x': 320, 'y': 163}, {'x': 320, 'y': 181}, {'x': 239, 'y': 181}], 'page': 1}, {'text': 'who', 'vertices': [{'x': 329, 'y': 163}, {'x': 362, 'y': 163}, {'x': 362, 'y': 181}, {'x': 329, 'y': 181}], 'page': 1}, {'text': 'fell', 'vertices': [{'x': 370, 'y': 164}, {'x': 410, 'y': 164}, {'x': 410, 'y': 181}, {'x': 370, 'y': 181}], 'page': 1}, {'text': 'in', 'vertices': [{'x': 410, 'y': 164}, {'x': 437, 'y': 164}, {'x': 437, 'y': 181}, {'x': 410, 'y': 181}], 'page': 1}, {'text': 'the', 'vertices': [{'x': 437, 'y': 164}, {'x': 470, 'y': 164}, {'x': 470, 'y': 181}, {'x': 437, 'y': 181}], 'page': 1}, {'text': 'Great', 'vertices': [{'x': 174, 'y': 204}, {'x': 234, 'y': 204}, {'x': 234, 'y': 221}, {'x': 174, 'y': 221}], 'page': 1}, {'text': 'War', 'vertices': [{'x': 234, 'y': 204}, {'x': 270, 'y': 204}, {'x': 270, 'y': 221}, {'x': 234, 'y': 221}], 'page': 1}]
```

Storage of Coordinates:

CSV Storage: After extracting the text and coordinates from the Vision API response, the script save this data to a CSV file. The CSV file includes columns for the text, vertices, and of each word.

	text	vertices	page
1	The	[{'x': 106, 'y': 163}, {'x':...	1
2	Men	[{'x': 154, 'y': 163}, {'x':...	1
3	of	[{'x': 212, 'y': 163}, {'x':...	1
4	Greenock	[{'x': 239, 'y': 163}, {'x':...	1
5	who	[{'x': 329, 'y': 163}, {'x':...	1
6	fell	[{'x': 370, 'y': 164}, {'x':...	1
7	in	[{'x': 410, 'y': 164}, {'x':...	1
8	the	[{'x': 437, 'y': 164}, {'x':...	1
9	Great	[{'x': 174, 'y': 204}, {'x':...	1
10	War	[{'x': 234, 'y': 204}, {'x':...	1

Dictionary Storage: Alternatively, the extracted data stored in a dictionary format within the script's memory. Each dictionary entry contains the text, vertices, and page of a word, making it easy to access and manipulate the data programmatically.

Search and Match:

Specifying Search Word: Users can input the search word they want to locate. This word is then used to search through the extracted data for a match.

```
pdf_path = "test.pdf"
search_word = "abbott, james"
draw_bounding_box(pdf_path, search_word)
```

Iterative Search: The script iterates through the stored data, comparing each word against the specified search word. Matching words are identified based on text similarity or exact matches, depending on the search criteria.

```
def find_word_coordinates(words, coordinates):
    word_coordinates = []
    # Split words on the basis of space, comma, or period if it's a string
    if isinstance(words, str):
        words = re.split(r'[ ,.]', words)
    for word in words:
        for coord in coordinates:
            if coord["text"].lower() == word.lower():
                word_coordinates.append({
                    "text": coord["text"],
                    "vertices": coord["vertices"],
                    "page": coord["page"]
                })
    return word_coordinates
```

Drawing Bounding Boxes:

PIL Library Usage: Upon finding the coordinates of the specified word, the script utilizes the Python Imaging Library (PIL) to draw bounding boxes around the word on each page of the PDF document.

Bounding Box Drawing: The PIL library provides functions to draw rectangles (bounding boxes) around specified coordinates on images. The script creates a new image for each page of the PDF, overlays bounding boxes around the identified word, and saves the modified images.

```

pil_image = Image.open(io.BytesIO(image))
draw = ImageDraw.Draw(pil_image)
for coord in word_coordinates:
    if isinstance(coord, dict) and coord['page'] == page_number + 1:
        vertices = coord['vertices']
        for i in range(4):
            draw.line([(vertices[i]['x'], vertices[i]['y']), (vertices[(i + 1) % 4]['x'], vertices[(i + 1) % 4]['y'])],
output_image_path = f"output_page_{page_number + 1}.jpg"
if any(isinstance(coord, dict) and coord['page'] == page_number + 1 for coord in word_coordinates):
    print(f"Saving bounding box on page {page_number + 1}...")
    pil_image.save(output_image_path)
    print(f"Bounding box saved as '{output_image_path}'.")

```

Visualization and Verification:

Image Output: The modified pages with bounding boxes are saved as images, allowing users to visually inspect the locations of the specified word within the PDF document.

Verification Process: Users can review the generated images to verify the accuracy of the bounding box placement and ensure that the specified word has been correctly identified on each page.

The Men of Greenock who fell in the Great War, 1914-1918.	
NAME.	UNIT.
ABBOTT, JAMES, Sergeant, - - - -	6th Royal Scots Fusiliers.
ADAMS, ADAM, Private, - - - -	2nd Highland Light Infantry.
ADAMS, ANDREW W., Driver, - - - -	Royal Army Service Corps.
ADAMS, DAVID, Private, - - - -	13th Royal Scots.
ADAMS, JOSEPH, Private, - - - -	20th King's London Regiment.
ADAMS, SIMON, Sapper, - - - -	Royal Engineers.
ADAMS, WILLIAM, Private, - - - -	11th Argyll and Sutherland Highlanders.
ADAMSON, JAMES, Private, - - - -	6th Connaught Rangers.
AINSLIE, JOHN E., 2nd Lieutenant, - - - -	9th Royal Scots.
AITKEN, ANDREW, 2nd Lieutenant, - - - -	South African Expeditionary Force.
AITKEN, CHARLES, Private, - - - -	5th Argyll and Sutherland Highlanders.
AITKEN, JAMES, Sergeant, - - - -	Royal Engineers.
AITKEN, ROBERT, Private, - - - -	2nd Gordon Highlanders.
AITKEN, WILLIAM, Private, - - - -	16th Royal Scots.
ALEXANDER, DONALD, Private, - - - -	11th Highland Light Infantry.
ALEXANDER, EDWARD, Private, - - - -	8th Argyll and Sutherland Highlanders.
ALEXANDER, EDWARD, Lance-Corporal, - - - -	7th King's Own Scottish Borderers.
ALEXANDER, HUMPHREY, Driver, - - - -	Royal Army Service Corps.
ALEXANDER, WILLIAM, Chief Engineer, - - - -	Transport Service.
ALLAN, GEORGE W., Private, - - - -	1st Highland Light Infantry.
ALLAN, GORDON S., Private, - - - -	8th Seaforth Highlanders.
ALLAN, JOHN, Lance-Corporal, - - - -	3rd Seaforth Highlanders.
ALLAN, JOHN B., Corporal, - - - -	2nd Argyll and Sutherland Highlanders.
ALLAN, PETER R., Private, - - - -	South African Expeditionary Force.
ALLEN, FRANK D., Private, - - - -	5th Argyll and Sutherland Highlanders.
ALLEN, THOMAS H., Lance-Corporal, - - - -	5th Argyll and Sutherland Highlanders.
ALLISON, NEIL, Private, - - - -	3rd Seaforth Highlanders.
ALLISON, SAMUEL, Gunner, - - - -	Royal Field Artillery.
ANDERSON, ARCHIBALD McC., - - - -	Transport Service.
ANDERSON, GEORGE A., Lieutenant, - - - -	6th Black Watch.
ANDERSON, GRAEME S., Corporal, - - - -	8th Seaforth Highlanders.

Usage

1. Open the coordinates.ipynb notebook in your Jupyter environment.
2. Provide the PDF file path and the search word.
3. Execute the script to visualize the bounding boxes around the specified word in the PDF document.
4. Check the generated images with bounding boxes to verify the locations of the search word.