Resume Parser

Industry Type: Human Resources/Recruitment

Project Title: Resume Parsing and Candidate Profile Synthesis

Problem Statement/Opportunity:
The manual process of parsing resumes and extracting relevant information for recruitment purposes is time-consuming and prone to errors. There is a need for an automated solution that can accurately parse resumes of various formats, extract key attributes such as roles, skills, period, education, companies, name, email, and phone number, and generate a concise synopsis of the candidate's profile. This will help recruiters in the IT industry to efficiently search and evaluate potential candidates.

Project Description:
The project aims to develop an automated resume parsing and candidate profile synthesis system using Natural Language Processing (NLP) techniques and Azure services. The system will provide a user-friendly GUI for recruiters to upload resumes, extract information, and interact with candidates. The key features of the project include:

- Resume Format Conversion: Implement a module to handle different resume formats such as .docx, .doc, .pdf, .txt, etc. Use Azure Cognitive Services (such as Azure Form Recognizer) to extract text content from these files.

- Information Extraction: Utilize NLP techniques, including Named Entity Recognition (NER) and rule-based approaches, to extract relevant attributes from the parsed text. Use Azure Cognitive Services like Azure Text Analytics or Azure Natural Language Understanding for NER tasks.

- Database Storage: Design a database schema to store the extracted information. Use Azure Cosmos DB, a NoSQL database service, for efficient storage and querying of candidate data.

- Synopsis Generation: Implement text summarization techniques, such as extractive or abstractive summarization, to generate a concise synopsis of the candidate's profile. Utilize Azure Text Analytics or Azure Language Understanding (LUIS) for text summarization tasks.

- Recruiters' GUI: Develop an intuitive GUI using Azure App Service or Azure Front Door that allows recruiters to upload resumes, initiate contact with candidates, search for specific skills or roles, and view synthesized profiles. Use Azure Active Directory for secure user authentication.

The project's purpose is to solve the problem of manual resume parsing and provide an automated solution to recruiters in the IT industry. The solution is clearly explained, outlining the steps involved in parsing resumes, extracting relevant information, and generating candidate synopses. The project's core functionality, including resume format conversion, information extraction, database storage, synopsis generation, and recruiters' GUI, is mapped to the problem statement and addresses the clear need for efficient candidate evaluation and search.

Primary Azure Technology:

Azure Form Recognizer: For extracting text content from different resume formats.

Here's a step-by-step guide on using Azure Cognitive Services, specifically Azure Form Recognizer, to extract text content from resume files:

Step 1: Set up an Azure Account
If you don't already have an Azure account, sign up for one at https://azure.microsoft.com/. You'll need an Azure subscription to access the required services.

Step 2: Create a Form Recognizer resource
Go to the Azure portal (https://portal.azure.com) and sign in with your Azure account credentials. Create a new resource by searching for "Form Recognizer" in the Azure Marketplace and selecting it from the results. Follow the prompts to create the resource in your preferred Azure subscription and resource group.

How to create a Form Recognizer resource - Azure Applied AI Services | Microsoft Learn

Step 3: Obtain the endpoint and access key
Once the Form Recognizer resource is created, navigate to the resource in the Azure portal. From the Overview page, note down the "Endpoint" URL and the "Key" value. You'll need these for authenticating and making requests to the service.

- After creating the Form Recognizer resource in the Azure portal, navigate to the Azure portal homepage (https://portal.azure.com) and sign in with your Azure account credentials.

- In the Azure portal, search for "Form Recognizer" in the search bar at the top. Select the "Form Recognizer" service from the search results.

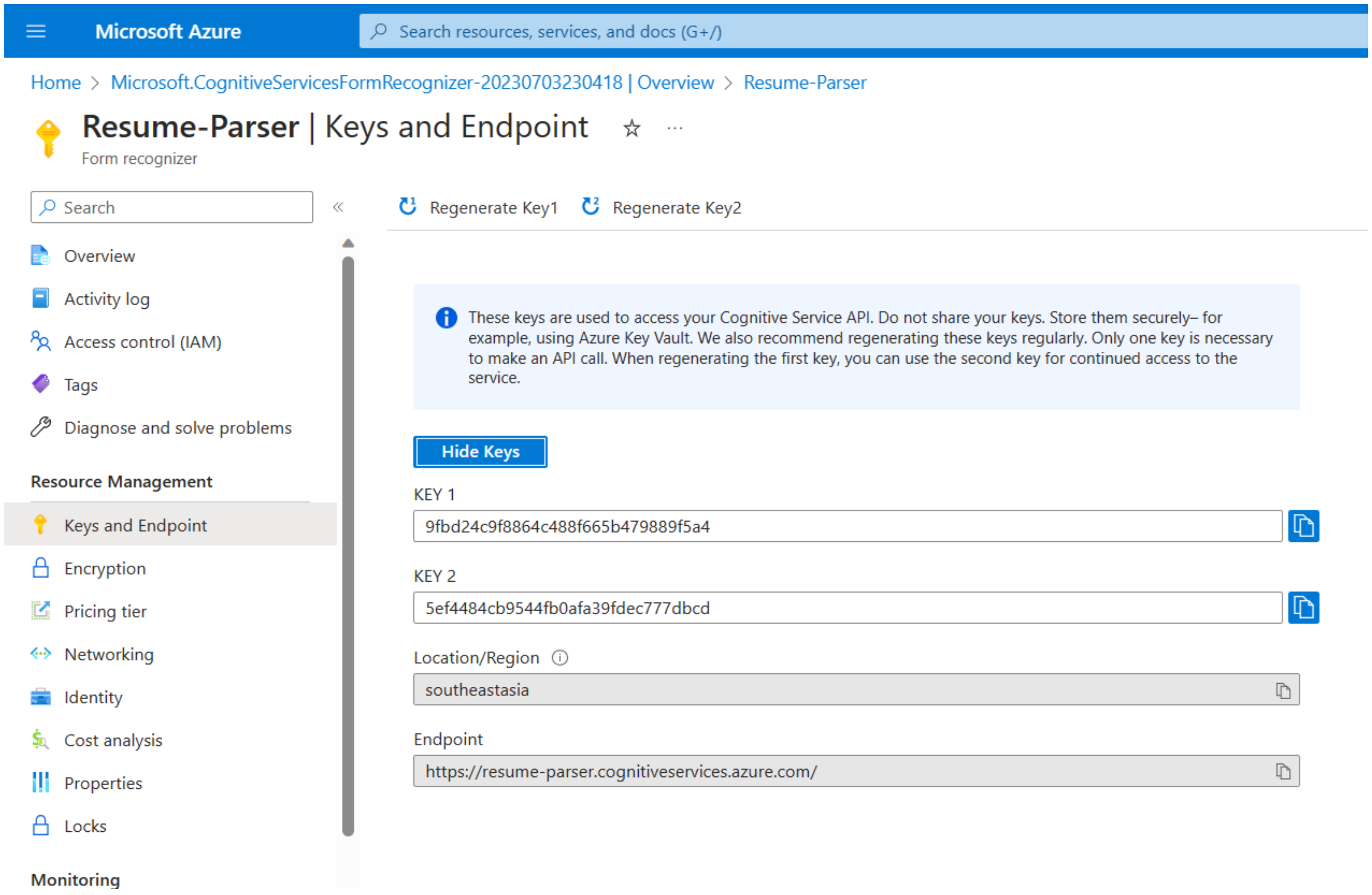
- You will be taken to the "Overview" page of the Form Recognizer resource. On this page, you will find important information about your resource, including the "Endpoint" URL and the "Key" value.

- The "Endpoint" URL is a unique URL that represents the location of your Form Recognizer resource. It typically follows the format: https://<resource-name>.cognitiveservices.azure.com/. Note down this URL as you'll use it to make API calls to the Form Recognizer service.

- The "Key" value is an access key that provides authentication and authorization to access the Form Recognizer service. It is used to secure your API calls. Click on the "Keys and Endpoint" tab on the left sidebar to view your access keys.

- On the "Keys and Endpoint" page, you will see two access keys listed: "Key1" and "Key2". These are both valid access keys for authenticating requests to the Form Recognizer service.

Note down either "Key1" or "Key2" (or both) as your access key(s). You'll need this key to include in the headers of your API requests to authenticate and authorize your calls to the Form Recognizer service.



These keys are used to access your Cognitive Service API. Do not share your keys. Store them securely– for example, using Azure Key Vault. We also recommend regenerating these keys regularly. Only one key is necessary to make an API call. When regenerating the first key, you can use the second key for continued access to the service.

KEY 1
9fbd24c9f8864c488f665b479889f5a4

KEY 2
5ef4484cb9544fb0afa39fdec777dbcd

Location/Region
southeastasia

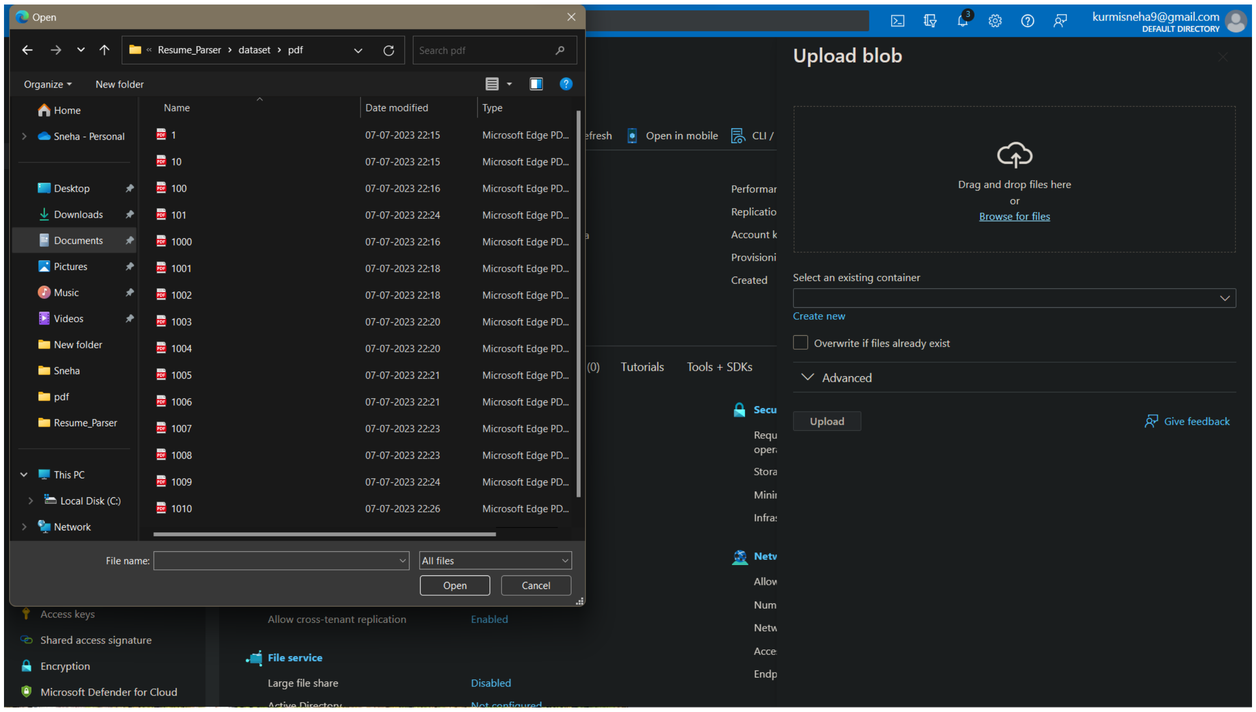
Endpoint
https://resume-parser.cognitiveservices.azure.com/

Fig: Keys and Endpoint

Step 4: Install Azure SDK or use REST APIs
To interact with Azure Form Recognizer, you can either use the Azure SDK for your preferred programming language (e.g., Python, C#, Java) or make HTTP requests directly to the REST API endpoints. Choose the approach that suits your project requirements and programming expertise.

Step 5: Authenticate with Azure Form Recognizer
To authenticate your requests, use the obtained endpoint URL and access key. Include the access key in the headers of your API requests as an "Ocp-Apim-Subscription-Key" parameter.

Step 6: Prepare and submit resume files for text extraction
Before sending a resume for text extraction, you may need to preprocess the files to remove any irrelevant content or perform format conversions if necessary (e.g., converting PDF to image or text). Make sure the resume file is in a supported format (e.g., PDF, JPEG, PNG).



Name	Date modified	Type
1	07-07-2023 22:15	Microsoft Edge PD...
10	07-07-2023 22:15	Microsoft Edge PD...
100	07-07-2023 22:16	Microsoft Edge PD...
101	07-07-2023 22:24	Microsoft Edge PD...
1000	07-07-2023 22:16	Microsoft Edge PD...
1001	07-07-2023 22:18	Microsoft Edge PD...
1002	07-07-2023 22:18	Microsoft Edge PD...
1003	07-07-2023 22:20	Microsoft Edge PD...
1004	07-07-2023 22:20	Microsoft Edge PD...
1005	07-07-2023 22:21	Microsoft Edge PD...
1006	07-07-2023 22:21	Microsoft Edge PD...
1007	07-07-2023 22:23	Microsoft Edge PD...
1008	07-07-2023 22:23	Microsoft Edge PD...
1009	07-07-2023 22:24	Microsoft Edge PD...
1010	07-07-2023 22:26	Microsoft Edge PD...

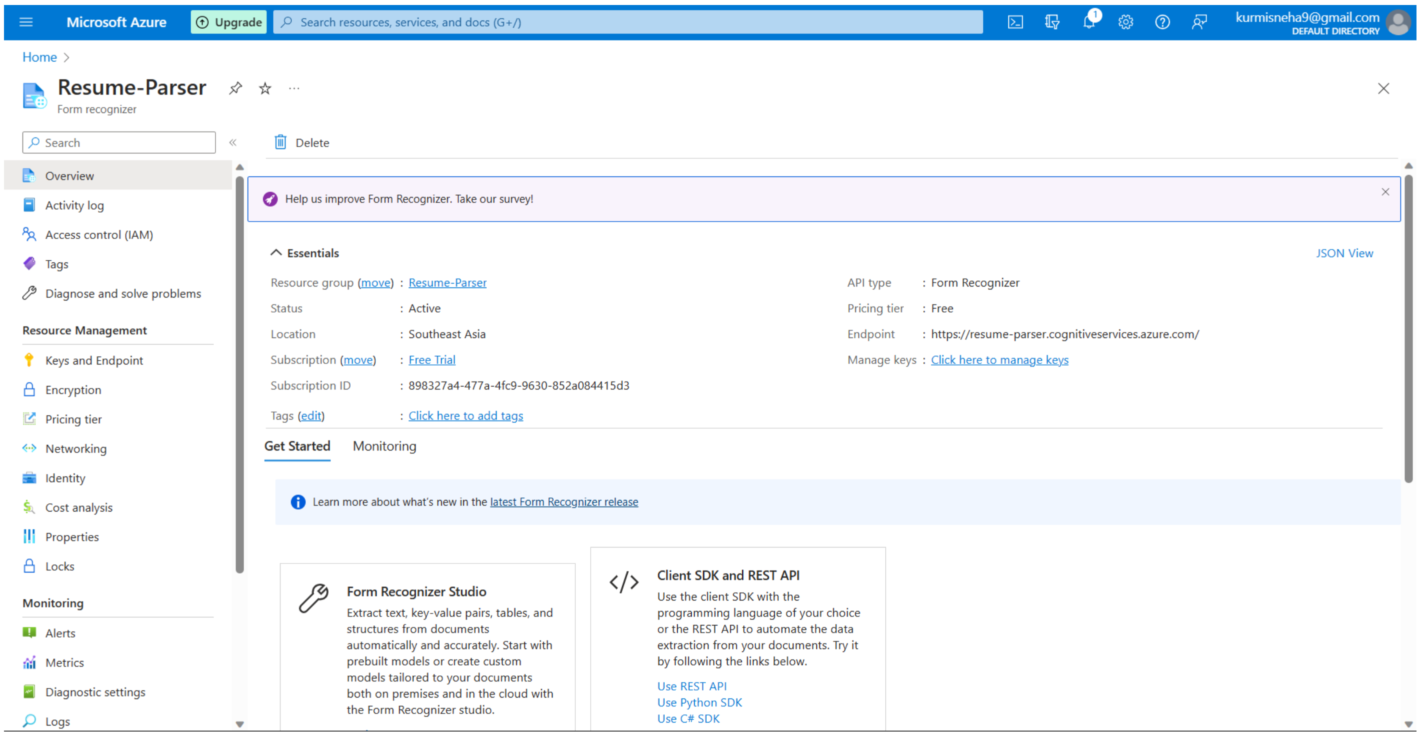
Allow cross-tenant replication Enabled
Large file share Disabled

Fig. Creating a storage account for custom extraction model and uploading the resumes there

Step 7: Make API calls for text extraction
Using the Azure SDK or sending HTTP requests, call the appropriate API endpoint to submit the resume files for text extraction. Use the "layout" or "prebuilt" mode in Form Recognizer, depending on your specific requirements. The API will analyze the document and return the extracted text content.

Step 8: Handle the response and extract relevant information

Receive the response from the API call, which will include the extracted text content from the resume. Parse and process the response to extract the desired information, such as the candidate's name, email, phone number, work experience, education, skills, etc.



Help us improve Form Recognizer. Take our survey!

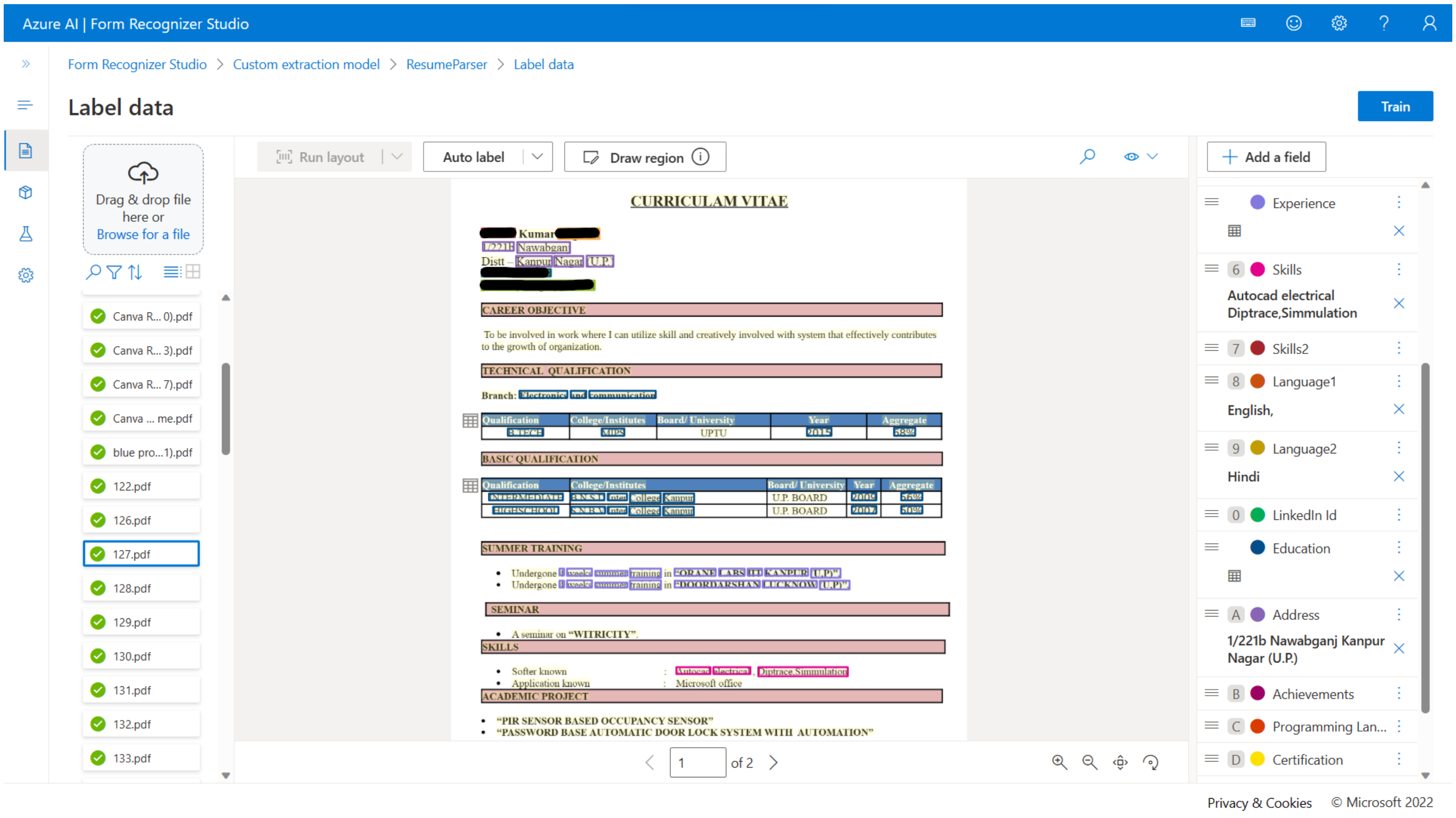
Resource group (move)	: Resume-Parser	API type	: Form Recognizer
Status	: Active	Pricing tier	: Free
Location	: Southeast Asia	Endpoint	: https://resume-parser.cognitiveservices.azure.com/
Subscription (move)	: Free Trial	Manage keys	: Click here to manage keys
Subscription ID	: 898327a4-477a-4fc9-9630-852a084415d3		
Tags (edit)	: Click here to add tags		

Learn more about what's new in the latest Form Recognizer release

Form Recognizer Studio
Extract text, key-value pairs, tables, and structures from documents automatically and accurately. Start with prebuilt models or create custom models tailored to your documents both on premises and in the cloud with the Form Recognizer studio.

Client SDK and REST API
Use the client SDK with the programming language of your choice or the REST API to automate the data extraction from your documents. Try it by following the links below.
Use REST API
Use Python SDK
Use C# SDK

Fig. Go to Form Recognizer Studio



Azure AI | Form Recognizer Studio

Form Recognizer Studio > Custom extraction model > ResumeParser > Label data

Label data

Train

Drag & drop file here or Browse for a file

- Canva R... 0).pdf
- Canva R... 3).pdf
- Canva R... 7).pdf
- Canva ... me.pdf
- blue pro...1).pdf
- 122.pdf
- 126.pdf
- 127.pdf
- 128.pdf
- 129.pdf
- 130.pdf
- 131.pdf
- 132.pdf
- 133.pdf

Run layout | Auto label | Draw region

CURRICULAM VITAE

[redacted] Kumar [redacted]
1/221b Nawabganj
Distt – Kanpur Nagar (U.P.)
[redacted]

CAREER OBJECTIVE

To be involved in work where I can utilize skill and creatively involved with system that effectively contributes to the growth of organization.

TECHNICAL QUALIFICATION

Branch: Electronics and communication

Qualification	College/Institutes	Board/ University	Year	Aggregate
B.TECH	MIPS	UPTU	2015	58%

BASIC QUALIFICATION

Qualification	College/Institutes	Board/ University	Year	Aggregate
INTERMEDIATE	B.N.S.D. Inter College Kanpur	U.P. BOARD	2009	56%
HIGHSCHOOL	S.N.B.V. Inter College Kanpur	U.P. BOARD	2007	50%

SUMMER TRAINING

- Undergone 4 weeks summer training in "ORANE LABS IIT KANPUR (U.P)"
- Undergone 4 weeks summer training in "DOORDARSHAN LUCKNOW (U.P)".

SEMINAR

- A seminar on "WITRICITY".

SKILLS

- Softer known : Autocad electrical , Diptrace.Simmulation
- Application known : Microsoft office

ACADEMIC PROJECT

- "PIR SENSOR BASED OCCUPANCY SENSOR"
- "PASSWORD BASE AUTOMATIC DOOR LOCK SYSTEM WITH AUTOMATION"

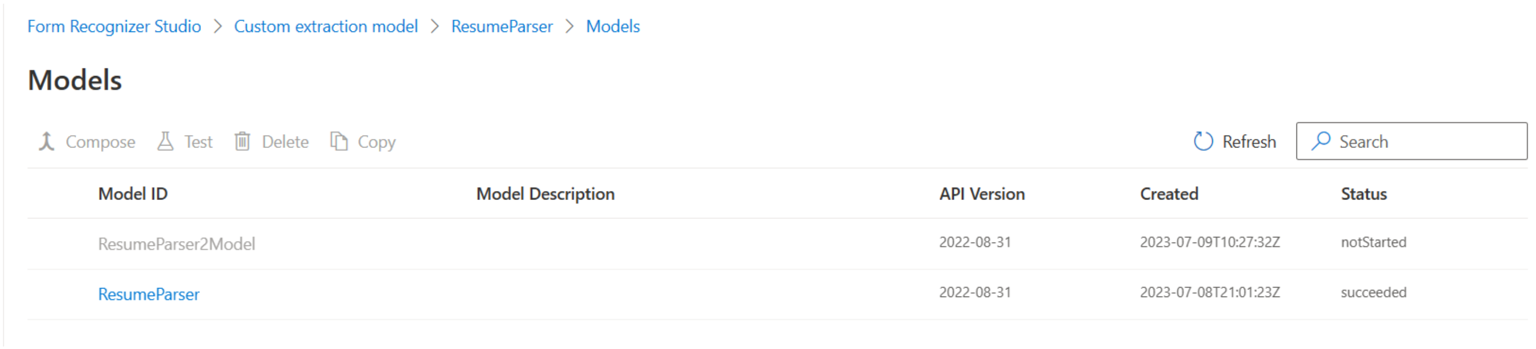
1 of 2

Add a field

Experience	▦
6 Skills	Autocad electrical Diptrace,Simmulation
7 Skills2	
8 Language1	English,
9 Language2	Hindi
0 LinkedIn Id	
Education	▦
A Address	1/221b Nawabganj Kanpur Nagar (U.P.)
B Achievements	
C Programming Lan...	
D Certification	

Privacy & Cookies © Microsoft 2022

Fig. Training custom extraction model



Form Recognizer Studio > Custom extraction model > ResumeParser > Models

Models

Compose Test Delete Copy Refresh Search

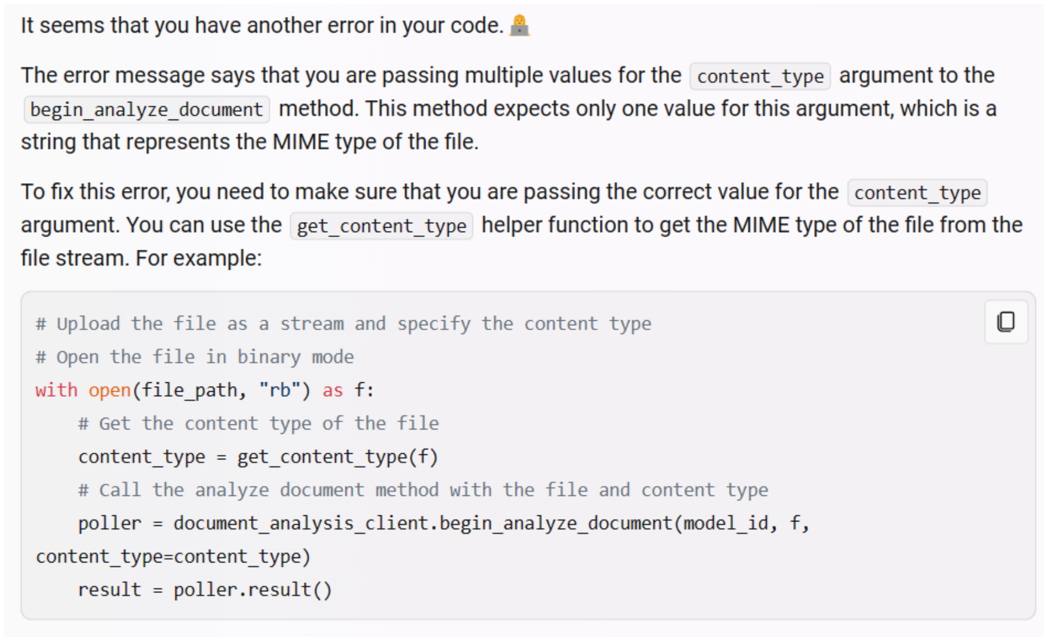
Model ID	Model Description	API Version	Created	Status
ResumeParser2Model		2022-08-31	2023-07-09T10:27:32Z	notStarted
ResumeParser		2022-08-31	2023-07-08T21:01:23Z	succeeded

[Form Recognizer Studio - Microsoft Azure](#)

Step 9: Integrate the extracted information into your solution
Take the extracted text content and integrate it into your solution's database or further processing logic. Store the relevant attributes in appropriate data structures or persist them in a database for later retrieval and analysis.

Code edit for website

So our main code takes file as document_url and since in the user interface user will be directly uploading the pdf or doc I tried to make the main code accept the pdf directly as a file.

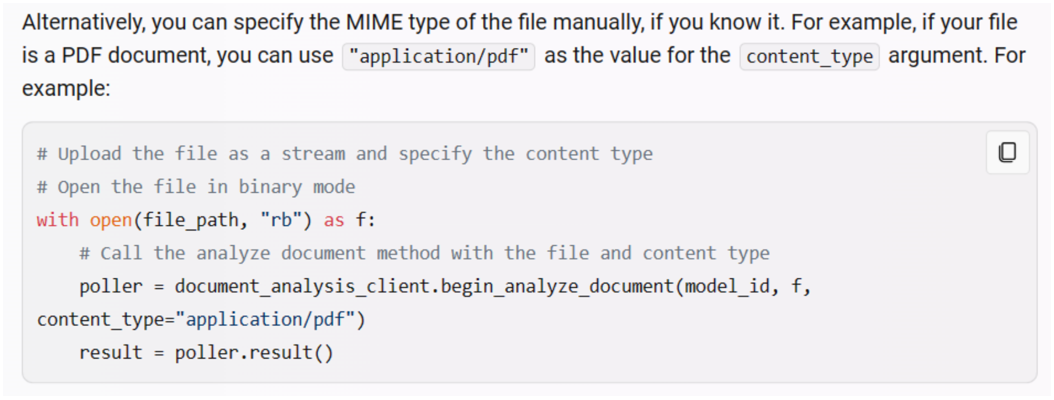


It seems that you have another error in your code. 🧑‍💻

The error message says that you are passing multiple values for the `content_type` argument to the `begin_analyze_document` method. This method expects only one value for this argument, which is a string that represents the MIME type of the file.

To fix this error, you need to make sure that you are passing the correct value for the `content_type` argument. You can use the `get_content_type` helper function to get the MIME type of the file from the file stream. For example:

```
# Upload the file as a stream and specify the content type
# Open the file in binary mode
with open(file_path, "rb") as f:
    # Get the content type of the file
    content_type = get_content_type(f)
    # Call the analyze document method with the file and content type
    poller = document_analysis_client.begin_analyze_document(model_id, f,
content_type=content_type)
    result = poller.result()
```

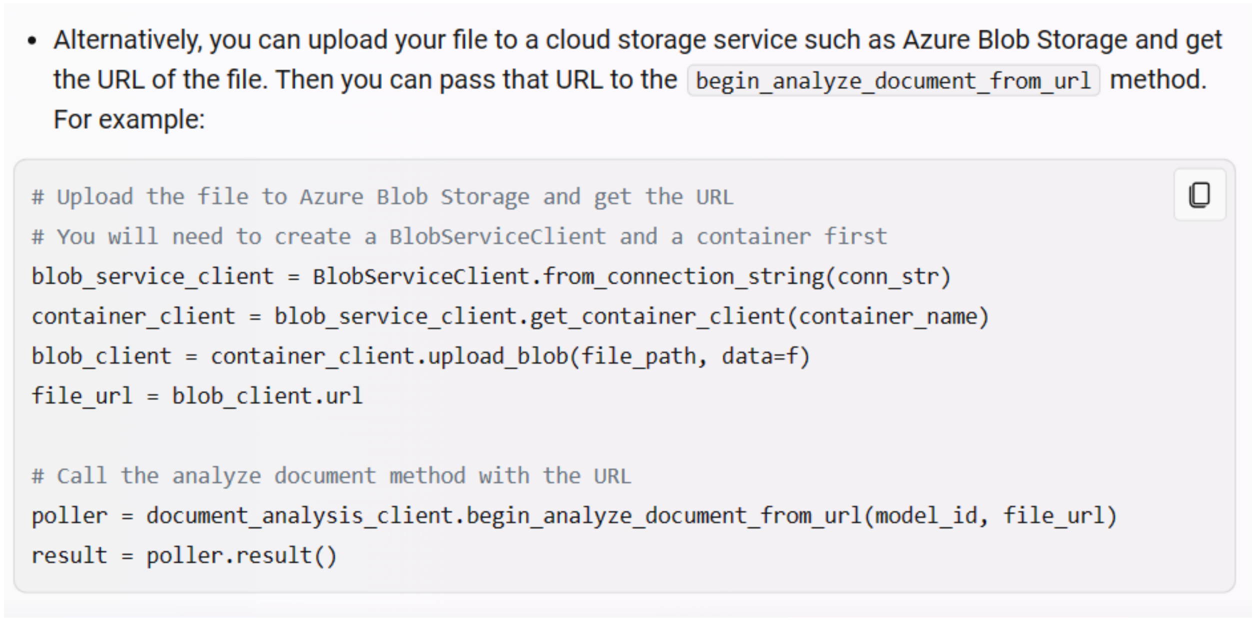


Alternatively, you can specify the MIME type of the file manually, if you know it. For example, if your file is a PDF document, you can use `"application/pdf"` as the value for the `content_type` argument. For example:

```
# Upload the file as a stream and specify the content type
# Open the file in binary mode
with open(file_path, "rb") as f:
    # Call the analyze document method with the file and content type
    poller = document_analysis_client.begin_analyze_document(model_id, f,
content_type="application/pdf")
    result = poller.result()
```

Both these methods didn't work out!

Then I used this method



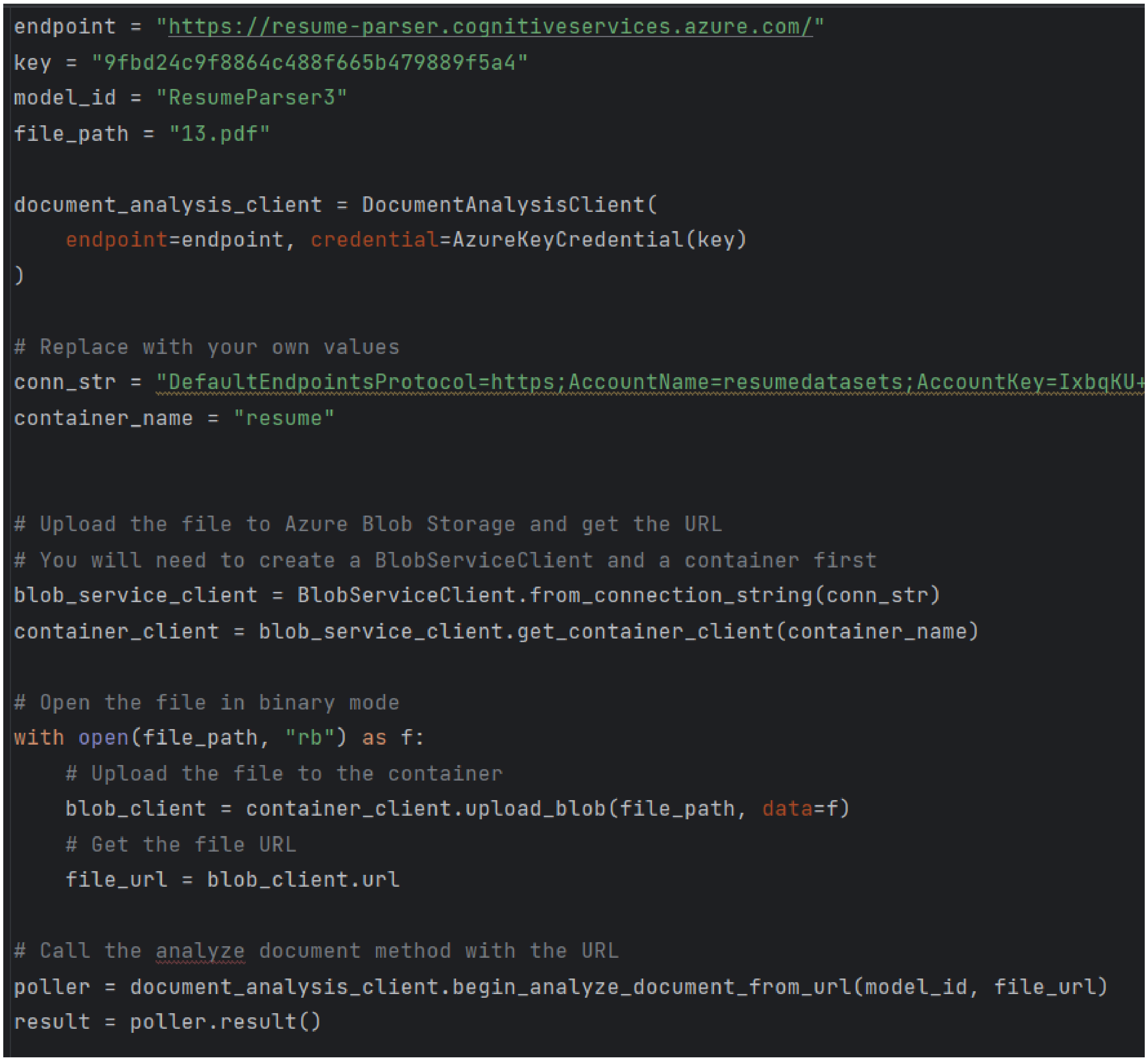
- Alternatively, you can upload your file to a cloud storage service such as Azure Blob Storage and get the URL of the file. Then you can pass that URL to the `begin_analyze_document_from_url` method. For example:

```
# Upload the file to Azure Blob Storage and get the URL
# You will need to create a BlobServiceClient and a container first
blob_service_client = BlobServiceClient.from_connection_string(conn_str)
container_client = blob_service_client.get_container_client(container_name)
blob_client = container_client.upload_blob(file_path, data=f)
file_url = blob_client.url

# Call the analyze document method with the URL
poller = document_analysis_client.begin_analyze_document_from_url(model_id, file_url)
result = poller.result()
```

I already had a container but since the file I was uploading was already in the container it was giving out error

So I deleted the whole container and created a new one and finally, it worked!
Here is the final code



```python
endpoint = "https://resume-parser.cognitiveservices.azure.com/"
key = "9fbd24c9f8864c488f665b479889f5a4"
model_id = "ResumeParser3"
file_path = "13.pdf"

document_analysis_client = DocumentAnalysisClient(
    endpoint=endpoint, credential=AzureKeyCredential(key)
)

# Replace with your own values
conn_str = "DefaultEndpointsProtocol=https;AccountName=resumedatasets;AccountKey=IxbqKU+
container_name = "resume"


# Upload the file to Azure Blob Storage and get the URL
# You will need to create a BlobServiceClient and a container first
blob_service_client = BlobServiceClient.from_connection_string(conn_str)
container_client = blob_service_client.get_container_client(container_name)

# Open the file in binary mode
with open(file_path, "rb") as f:
    # Upload the file to the container
    blob_client = container_client.upload_blob(file_path, data=f)
    # Get the file URL
    file_url = blob_client.url

# Call the analyze document method with the URL
poller = document_analysis_client.begin_analyze_document_from_url(model_id, file_url)
result = poller.result()
```