

Gen AI Project Using Llama 3.1 (LLM)

Cold Mail Generator

Technical Architecture

1. Extract text from career page - webBaseLoader from langchain
2. Pass into Llama LLM convert it into json format with (role, skills, description)
3. Send it to Chromadb (Vector storage)

Installation in colab

1. Pip3 install langchain
2. Pip3 install langchain groq

Step : 1

Import langchain_groq from chatgroq for using Llama LLM

```
import chatgroq from langchain_groq for accessing llama LLM it went to groq cloud and recieved the help of llama and got us the required answer
```

```
from langchain_groq import ChatGroq

llm = ChatGroq(
    groq_api_key = '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',
    model="llama-3.1-70b-versatile",
    temperature=0,
)

response = llm.invoke("Who is first person to go to moon?")
print(response)
```

```
content='The first person to walk on the Moon was Neil Armstrong. He stepped out of the lunar module Eagle and onto the Moon\'s surface on July 20, 1969, during'
```

Now the LLM (Llama model) can be used to invoke several tasks for the below process.

Step : 2

Chromadb - A Python library for managing embeddings for dense vector representations

Install Chromadb - **!pip install chromadb**

Code:

```
Client = chromadb.client()
Collect = client.create_collection(name = "my_collect")
```

Explanation: The above code calls the chromadb for data embeddings, which can be used to create collection, insert data, and query results. The collection often represents related data within the vector storage space.

```
collect.add(
    Document = [
        "This is a document about new york",
        "This is a document about London"
    ],
    Ids = ['id1', 'id2']
)
```

Explanation: It adds data in document format and converted into embeddings before storage and each document data provided with unique identifiers

Retrieving:

```
All_collection = collect.get()
All_collection
```

Output:

```
{'ids': ['id1', 'id2'],
'embeddings': None,
'documents': ['This document is about New York',
'This document is about London'],
'uris': None,
'data': None,
'metadatas': [None, None],
'included': [<IncludeEnum.documents: 'documents'>,
<IncludeEnum.metadatas: 'metadatas'>]}
```

Querying: To match texts for similar results

```
Result = collect.query(  
    Query_texts = ["This is brooklyn bridge"]  
    N_results = 2  
)  
Result
```

Use cases of Chromadb:

1. Semantic Search - store text embeddings for querying similar results
2. AI ML - manage embeddings for machine learning model

Vector database

Process

Firstly we'll generate embeddings using openai API -> those will give vector values -> which can be matched with the one stored in databases by earlier processes using cosine similarity technique.

Instead of linear search for millions of vector values, we can use hashing and indexing called LSH Function

Locality sensitive hashing - search through bucket using same hashing function

Why use a vector database rather than a normal DB?

- It uses cosine similarity function as a use case of semantic search where the search is based on euclidean distance of vector information in chromadb

Step : 3

WebBaseLoader

It will extract the data or scripts from the link - WebScrapping

Code:

```
From langchain.community.document_loaders import WebBaseLoader  
Loader = WebBaseLoader("https://jobs.nike.com/ref-6095")  
Page_data = loader.load().pop().page_content  
print(page_data)
```

We can use prompt templates to extract data in particular format and the concise and precise way we want

Code:

From langchain_core.prompts import PromptTemplate

```
Prompt_extract = promptTemplate.from_template(
    """
        SCRAPPED TEXT FROM WEBSITE
        {page_data}
        The scrapped text is from careers page, your job is to
        Extract job postings and roles, skills and return them in
        Json format in containers
        Following keys : 'role', 'skills', 'description'
        ### valid_json(NO PREAMBLE):
    """
)
```

```
Chain_extract = prompt_extract | llm
Res = chain_extract.invoke({"page_data": page_data})
print(res.content)
```

Output:

```
```json
{
 "role": "Sr Designer, Digital Product Design",
 "experience": "5+ years of relevant experience with end-to-end digital
product design",
 "skills": [
 "UX/UI design",
 "human-computer interaction",
 "e-commerce",
 "UX best practices",
 "product thinking",
 "interaction design",
 "execution",
 "Figma",
 "Adobe CC",
 "Principle",
 "Keynote"
],
}
```

```
"description": "Designing incredibly polished work that represents the
Nike brand and product experience through visual comps, wireframes, flows
and prototypes. Owning multiple product features for medium-to-large
projects. Rapidly iterating and refining design concepts—from initial
briefing through final delivery. Working closely and collaboratively with
a cross-functional team of producers, product managers, project managers,
engineers, researchers and fellow designers."
}
```

The Type of above output is of 'str', we need to convert them into json format using json parser from langchain

**Code:**

```
From langchain_core.output_parsers import JsonOutputParser
Json_parser = JsonOutputParser()
Json_res = json_parser(content.res)
```

**Step : 4**

Task : You will be given a list of project and their corresponding URL, when a particular job posting is mentioned we retrieve the key features of skills used in the projects and their url to be inserted into the cold email for particular client

Get that data into your colab environment and iterate them one by one to store them into chromadb.

**Code:**

```
Import pandas as pd
df = pd.read_csv("my_portfolio.csv")
```

**> Chromadb Code**

**Import chromadb**

```
Client = chromadb.PersistentClient('vectorstore')
Collection = client.get_or_create_collection(name = "portfolio")
```

Note: We use persistent Client in above code in difference from the last use of only chromadb.client. Because in persistent client it will create database in disk (folder) so that next time when we retrieve data it is on disk

```

If not collection.count():
 for _, rows in df.iterrows():
 collection.add(documents=row["Techstack"],
 metadata={ "links": row["links"] },
 ids=[str(uuid.uuid4())])

```

```

Link = collection.query(query.texts=['react'], n_results=2).get('metadata', [])
Link

```

### Output:

```

[[{'links': 'https://example.com/react-portfolio'},
 {'links': 'https://example.com/react-native-portfolio'}]]

```

It gives the links matching the skills = 'react'

### Step: 4

Create prompt template for forming an email

We now pass 2 prompts {job} and {link list} into our LLM so that it generates cold email with the format we specified

```

prompt_email = PromptTemplate.from_template(
 """
 ### JOB DESCRIPTION:
 {job_description}

 ### INSTRUCTION:
 You are Mohan, a business development executive at AtliQ. AtliQ is an AI & Software
 Consulting company dedicated to facilitating
 the seamless integration of business processes through automated tools.
 Over our experience, we have empowered numerous enterprises with tailored solutions,
 fostering scalability,
 process optimization, cost reduction, and heightened overall efficiency.
 Your job is to write a cold email to the client regarding the job mentioned above describing
 the capability of AtliQ
 in fulfilling their needs.
 Also add the most relevant ones from the following links to showcase Atliq's portfolio:
 {link_list}
 Remember you are Mohan, BDE at AtliQ.
 Do not provide a preamble.
 ### EMAIL (NO PREAMBLE):
 """

```

)

```
chain_email = prompt_email | llm
res = chain_email.invoke({"job_description": str(job), "link_list": links})
print(res.content)
```

### Output:

### Cold email

```
Subject: Expert Digital Product Design Solutions for Nike

Dear Hiring Manager,

I came across the job description for a Sr Designer, Digital Product Design at Nike, and I'm excited to introduce AtliQ, an
At AtliQ, we specialize in UX/UI design, human-computer interaction, e-commerce, and UX best practices, aligning perfectly w
Our portfolio showcases our expertise in designing and developing digital products, including:

* React-based solutions: https://example.com/react-portfolio
* React Native-based solutions: https://example.com/react-native-portfolio


These examples demonstrate our capability in delivering scalable, optimized, and efficient digital products that meet the hi
I'd love to discuss how AtliQ can support Nike's digital product design requirements. Please let me know if you're intereste
Best regards,

Mohan
Business Development Executive
AtliQ
```

Use Python and streamlit library for UI

Result:

Deploy

 **Cold Mail Generator**

Enter a URL:

<https://www.linkedin.com/jobs/collections/recommended/?currentJobId=4109918108>

Submit

Subject: Expert Software Engineering Solutions for Your Business

Dear Hiring Manager,

I came across your job posting for a Software Engineer with 2-5 years of experience in Python, Java, and C++. I'd like to introduce you to AtliQ, an AI & Software Consulting
At AtliQ, we have a proven track record of empowering enterprises with scalable, process-optimized, and cost-effective solutions. Our team of expert software engineers has e
Our portfolio showcases our capabilities in software development, including:
- Machine learning and Python-based solutions: <https://example.com/ml-python-portfolio>
- Java-based solutions: <https://example.com/java-portfolio>
- Python-based solutions: <https://example.com/python-portfolio>

We'd be happy to discuss how our expertise can align with your requirements and provide a customized solution for your business. Please feel free to reply to this email or s
Best regards,
Mohan
Business Development Executive
AtliQ