

گزارش کار پروژه چت بات

هدف از این پروژه ایجاد چت بات برای پاسخ به سوالات کاربران با توجه به دو دیتاست financial و customer support است. با توجه به متن تسک ارائه شده چت بات توسعه داده شده باید ویژگی‌های زیر را داشته باشد:

- 1- از مدل GPT 3.5 Turbo برای پاسخ دهی استفاده شود
- 2- با فریمورک LangChain پیاده سازی کنید
- 3- در طول مکالمه از هیستوری چت استفاده کند
- 4- کاهش اثر Hallucination با راه حلی غیر از fine tuning
- 5- سوالات مالی و سوالات تخصصی توسط چت بات مربوطه پاسخ داده شود

در ادامه گزارش تمامی ویژگی‌های مذکور بررسی میشوند.

شرح الگوریتم استفاده شده

مدل‌های زبانی بزرگ، که با حجم عظیمی از داده‌های متنی آموزش دیده اند، قادر به تولید متنی هستند که از نظر ظاهری با متن نوشته شده توسط انسان قابل تشخیص نیست. با این حال، این مدل‌ها نیز مستعد ایجاد توهم هستند. توهم در مدل‌های زبانی به معنای تولید متنی است که واقعی یا دقیق نیست. برای جلوگیری از توهم در مدل‌های زبانی روش‌های متعددی پیشنهاد میشود از جمله فاین تیون کردن مدل، قابلیت RAG (Retrieval Augmented Generation) و غیره.

در این پروژه به منظور جلوگیری از توهم از ویژگی RAG با بهره‌گیری از Elastic search استفاده شده است و همچنین برای پیاده سازی آن از فریمورک LangChain استفاده شده است.

معماری سیستم RAG به این صورت است که در ابتدا تمامی اسناد و دیتاست‌ها که در اینجا فایل CSV بود به قسمت‌های مشخصی که به آن chunk گفته میشود تقسیم شده و سپس embed میشوند و در یک vector space ذخیره میشوند. Vector space های زیادی وجود دارند از جمله chromaDB, FAISS و یا الستیک سرچ (در این پروژه از الستیک سرچ به منظور سرعت واکنشی بالاتر استفاده شده است). پس از ذخیره سازی دیتاست درون vector space، شباهت پرسش کاربر یا وکتورهای embed ذخیره شده از طریق روش‌های مختلف به عنوان مثال semantic Search و یا TF IDF بررسی شده و چانک‌های مربوطه واکنشی میشوند. پس از آن Chain مربوطه با توجه به Prompt template و LLM مورد نظر (در این پروژه از GPT 3.5 turbo استفاده شده است) ساخته میشود. در این Chain سوال به همراه context که همان چانک واکنشی شده است به مدل LLM داده میشود و از LLM خواسته میشود تا با توجه به این کانتکست جواب سوال را بیابد.

در ادامه به شرح جزئی‌تر کد پرداخته میشود.

شرح کد استفاده شده

در تعریف تسک بیان شده است که از دو چت بات استفاده شود و با توجه به پرسش کاربر تشخیص داده شود که مربوط به کدام چت بات است. در این پروژه از آنجا که تمامی دیتاست درون الستیک سرچ ذخیره میشوند نیازی به ایجاد دو چت بات نیست و در

واقع دو دیتاست با یکدیگر تلفیق شده اند. اگر بخواهیم از دو چت بات مجزا استفاده کنیم overhead اضافه بر روی برنامه تحمیل میکنیم. ولی اگر بخواهیم با توجه به پرسش کاربر تشخیص دهیم که از چه چت باتی استفاده شود تنها کافی است از روش های classification به منظور تشخیص نوع سوال کاربر استفاده شود ولی در این پروژه پس از تلفیق و join دو دیتاست نیازی به انجام این کار نیست.

لذا در ابتدا دو دیتاست CSV خوانده شده و درون دیتافریم pandas ذخیره شده و سپس تلفیق گشتند و نتیجه به صورت یک فایل json ذخیره شد. قطعه کد زیر نمایانگر همین موضوع است.

```
def load_data_to_json():
    df1 = pd.read_csv ('../dataset/customer_support.csv',encoding='utf-8',sep = ",", header = 0, index_col = False)
    df2 = pd.read_csv ('../dataset/financial.csv',encoding='utf-8',sep = ",", header = 0, index_col = False)

    df = pd.concat([df1, df2], axis=0)

    print(df)
    df.to_json ('joined_datasets.json',orient = "records", date_format = "epoch", double_precision = 10, force_ascii = True, date_unit
```

سپس در تابع index_data فایل json که تلفیق دو دیتاست بود خوانده شده و به منظور انجام RAG به چانک های مشخص تقسیم شده و به صورت امبد شده درون الستیک سرچ ذخیره میشود. قطعه کد زیر نمایانگر همین مورد است. همان طور که در قطعه کد زیر مشخص شده است 6 چانک مرتبط با پرسش کاربر واکنشی میشوند.

```
def index_data(data_url):
    metadata = []
    content = []

    f = open(data_url)
    workplace_docs = json.load(f)
    for doc in workplace_docs:
        content.append(doc["fact"])
        metadata.append({
            "question": doc["question"],
        })

    text_splitter = CharacterTextSplitter(chunk_size=100, chunk_overlap=0)
    docs = text_splitter.create_documents(content, metadatas=metadata)
    embeddings = OpenAIEmbeddings(openai_api_key=openai_api_key)
    es = ElasticVectorSearch.from_documents(
        docs,
        elasticsearch_url=url,
        index_name=elastic_index_name,
        embedding=embeddings
    )
    retriever = es.as_retriever(search_kwargs={"k": 6})
    return retriever,es
```

تصاویر زیر نشان دهنده ی اسناد ذخیره شده در الستیک سرچ است.

```
Console Search Profiler Grok Debugger Painless Lab BETA

History Settings Variables Help

1 GET _search
2 {
3   "query": {
4     "match_all": {}
5   }
6 }
7
8
9 PUT /byte-discuss-langchain-rag
10
11
12 GET /_cat/indices?v
13
14 GET /byte-discuss-langchain-rag/_search
15 {
16   "query": {
17     "match_all": {}
18   }
19 }
20
21
22
23 DELETE /byte-discuss-langchain-rag
24
25

1 {
2   "took": 5,
3   "timed_out": false,
4   "_shards": {
5     "total": 1,
6     "successful": 1,
7     "skipped": 0,
8     "failed": 0
9   },
10  "hits": {
11    "total": {
12      "value": 231,
13      "relation": "eq"
14    },
15    "max_score": 1,
16    "hits": [
17      {
18        "_index": "byte-discuss-langchain-rag",
19        "_id": "5355325c-f921-4c52-aa53-cf57b6716cea",
20        "_score": 1,
21        "_source": {
22          "vector": [
23            0.0006800651816084165,
24            0.01021841354503946,
25            -0.012342576128479018,
26            -0.04124307262408543,
27            -0.014328207260851303,
28            -0.001340795663098712,
29            -0.032640869983238806,
30            -0.015713511090185766,
31            0.008054668816185998,
32            -0.025701057648174857,
33            0.000000000000000000
34          ]
35        }
36      }
37    ]
38  }
39 }
```

```
"hits": [
  {
    "_index": "byte-discuss-langchain-rag",
    "_id": "5355325c-f921-4c52-aa53-cf57b6716cea",
    "_score": 1,
    "_source": {
      "vector": [0.0006800651816084165,
0.01021841354503946,
-0.012342576128479018,
-0.04124307262408543,
-0.014328207260851303,
-0.001340795663098712,
-0.032640869983238806,
-0.015713511090185766,
0.008054668816185998,
-0.025701057648174857,
0.000000000000000000],
      "text": "خیر، هاست های ما همه لینوکسی هستند، میتونید سرور ویندوزی تهیه کنید",
      "metadata": {
        "question": "هاست ویندوز دارید؟"
      }
    }
  },
  {
    "_index": "byte-discuss-langchain-rag",
    "_id": "4c106b79-c58c-42fb-8441-61da9c6aad12",
    "_score": 1,
    "_source": {
      "vector": [0.0006800651816084165,
0.01021841354503946,
-0.012342576128479018,
-0.04124307262408543,
-0.014328207260851303,
-0.001340795663098712,
-0.032640869983238806,
-0.015713511090185766,
0.008054668816185998,
-0.025701057648174857,
0.000000000000000000],
      "text": "روى هاست هاى پربازديد و اختصاصى امكان پذير است SSH دسترسى",
      "metadata": {
        "question": "وجود داره؟ ssh روى کدام يك از هاست هاى پارس پک دسترسى"
      }
    }
  },
  {
    "_index": "byte-discuss-langchain-rag",
    "_id": "c80f0c0e-f616-42de-8a7c-7cfb918fe3b7",
    "_score": 1,
    "_source": {
      "vector": [0.0006800651816084165,
0.01021841354503946,
-0.012342576128479018,
-0.04124307262408543,
-0.014328207260851303,
-0.001340795663098712,
-0.032640869983238806,
-0.015713511090185766,
0.008054668816185998,
-0.025701057648174857,
0.000000000000000000],
      "text": "پس از ثبت نام از داخل پنل کاربری، قسمت هاست ابری، هاست رایگان قابل سفارش است",
      "metadata": {
        "question": "پس از ثبت نام از داخل پنل کاربری، قسمت هاست ابری، هاست رایگان قابل سفارش است"
      }
    }
  }
]
```

در تابع elastic_chat، chain مربوطه با توجه به Prompt template و LLM ساخته شده و جواب سوال کاربر با توجه به متن واکشی شده، استخراج می شود.

```

def elastic_chat(question):
    r = es.similarity_search(question)
    print(showResults(r))
    template = """Answer in Persian: Answer the question based only on
the following context:
{context}
Question: {question}
"""
    prompt = ChatPromptTemplate.from_template(template)

    chain = (
        {"context": retriever, "question": RunnablePassthrough()}
        | prompt
        | ChatOpenAI()
        | StrOutputParser()
    )
    response = chain.invoke(question)
    return response

```

به کمک Flask، api مربوط به این تابع نیز نوشته شده است.

```

@app.route('/elastic', methods=['GET'])
def conversational_chat():
    # try:
        question = str(request.args.get('question'))
        print(question)
        print('\n')
        result = elastic_chat(question)
        print(result)
        final_result = {"result":result}
        return final_result
    # except:
    #     return jsonify({'result':str("An unexpected error happened")})

```

تصاویر زیر نمونه ای از نتایج را نشان میدهد.

The screenshot displays the Postman application interface. The top navigation bar includes 'Home', 'Workspaces', and 'API Network'. The main workspace is titled 'Personal Workspace' and shows a collection of requests. The selected request is a GET request to 'http://localhost:5001/elastic?question=چگونه میتوانم حق اشتراک خود را پرداخت کنم'. The 'Query Params' section shows a table with two entries: 'Key' and 'question', both with values. The 'Body' tab is selected, showing a JSON response in 'Pretty' format. The response is a 200 OK status with a body containing a Persian message about PayPal and subscription payments.

GET http://localhost:5001/elastic?question=چگونه میتوانم حق اشتراک خود را پرداخت کنم

Params • Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description
question	چگونه میتوانم حق اشتراک خود را پرداخت کنم	
Key	Value	Description

Body Cookies Headers (5) Test Results 200 OK 5.79 s 783 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "result": "شما می‌توانید از طریق کارت‌های اعتباری، حواله بانکی یا سیستم‌های پرداخت آنلاین نظیر PayPal حق اشتراک خود را پرداخت نمایید."
3 }
```

Home Workspaces API Network Search Postman Invite Upgrade

Personal Workspace New Import Overview GET http://156.253.5.103:50 GET http://localhost:5001/el. No Environment

Collections + Environments History

My first collection
First folder inside collection
Second folder inside collection

Create a collection for your requests

A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.

Create Collection

http://localhost:5001/elastic?question=آیا هاست ویندوز ارائه میکنید

GET http://localhost:5001/elastic?question=آیا هاست ویندوز ارائه میکنید Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description
question	آیا هاست ویندوز ارائه میکنید	
Key	Value	Description

Body Cookies Headers (5) Test Results 200 OK 4.75 s 529 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "result": "خیر، هاست های ما همه لینوکسی هستند، نمیتوانید سرور ویندوزی تهیه کنید."
3 }
```

Online Find and replace Console Postbot Runner Start Proxy Cookies Trash

Home Workspaces API Network Search Postman Invite Upgrade

Personal Workspace New Import Overview GET http://156.253.5.103:50 GET http://localhost:5001/el. No Environment

Collections + Environments History

My first collection
First folder inside collection
Second folder inside collection

Create a collection for your requests

A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.

Create Collection

http://localhost:5001/elastic?question=چگونه میتوانم از سرور تست استفاده کنم

GET http://localhost:5001/elastic?question=چگونه میتوانم از سرور تست استفاده کنم Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description
question	چگونه میتوانم از سرور تست استفاده کنم	
Key	Value	Description

Body Cookies Headers (5) Test Results 200 OK 4.55 s 542 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "result": "شما میتوانید از بخش سرور/ایری پتل کاربری اقدام به ایجاد سرور تست نمایید."
3 }
```

Online Find and replace Console Postbot Runner Start Proxy Cookies Trash

پیاده‌سازی ویژگی حفظ تاریخچه گفتگو

به منظور حفظ تاریخچه ی گفتگو از دیتابیس PostgreSQL و همچنین استفاده از پرسش و پاسخ های قبلی در prompt ورودی به مدل استفاده شده است. برای همین منظور یک table با نام chat در postgresql ساخته شده است که شامل دو فیلد user_id و همچنین chat_history است. در این پروژه تنها از یک یوزر به نام Farzaneh استفاده شده است.

```
CREATE TABLE chat( pid SERIAL PRIMARY KEY, user_id CHARACTER VARYING, chat_history CHARACTER VARYING);  
-  
select * from chat  
SELECT chat_history FROM chat WHERE user_id = 'farzaneh'
```

همچنین از prompt template زیر استفاده شده است.

Please just answer the question in Persian

Start of Conversation History

User: پایتخت ایران کجا است ؟

Chatbot: پایتخت ایران تهران است.

User: جمعیت این شهر چقدر است ؟

Chatbot: جمعیت تهران، پایتخت ایران، تقریباً 9 میلیون نفر است.

[End of Conversation History]

[Current Interaction]

User: سوال آخریم چیه

در تابع elastic_chat_history در ابتدا با توجه به user_id مقدار chat history واکشی شده و Prompt مورد نظر

ساخته میشود. قطعه کد زیر عملکرد این فانکشن را نشان میدهد.

```

def chat_history(question):
    history = get_data_from_db("farzaneh")
    primary_question = question
    if len(history) > 0 :
        question = history[-1] + "Current Interaction User :" + primary_question
    else:
        question = "User :" + primary_question

    print(question)

    r = es.similarity_search(question)
    print(showResults(r))
    template = """Answer in Persian: Answer the question based only on the following context:
    {context}
    Question: {question}
    """
    prompt = ChatPromptTemplate.from_template(template)

    chain = (
        {"context": retriever, "question": RunnablePassthrough()}
        | prompt
        | ChatOpenAI()
        | StrOutputParser()
    )
    response = chain.invoke(question)
    question = question.replace("Start of Conversation History", "")
    question = question.replace("Current Interaction", "")
    question = question.replace("End of Conversation History", "")

```

به منظور بررسی نحوه‌ی عملکرد این تابع به ترتیب سوالات زیر از چت بات پرسیده شد و نتایج مرتبط دریافت گردید. تصاویر زیر این پرسش و پاسخ را نمایش میدهد

- 1- آیا میتوانم برای سایت CDN فعال کنم؟
- 2- چگونه فعال کنم؟
- 3- مراحلش رو بگو

http://localhost:5001/history?question=فعال كنم؟ CDN آيا ميتونم براي سايتم

Save

</>

GET

http://localhost:5001/history?question=فعال كنم؟ CDN آيا ميتونم براي سايتم

Send

ParamsAuthorizationHeaders (6)BodyPre-request ScriptTestsSettingsCookies

Query Params

<input checked="" type="checkbox"/> Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/> question	فعال كنم؟ CDN آيا ميتونم براي سايتم			
Key	Value	Description		

BodyCookiesHeaders (5)Test Results

Status: 200 OKTime: 3.88 sSize: 971 BSave as example

PrettyRawPreviewVisualizeJSON

```
1 {
2   "result": {
3     "final response": "استفاده كنيد CDN بله، ميتوانيد براي سايت خود از سرويس",
4     "prompt": "Start of Conversation History  User : فعال كنم؟ CDN آيا ميتونم براي سايتم , chatbot: ميتوانيد براي سايت خود از سرويس .End of
5       Conversation History"
6   }
}
```

http://localhost:5001/history?question=چگونه فعال كنم؟

Save

</>

GET

http://localhost:5001/history?question=چگونه فعال كنم؟

Send

ParamsAuthorizationHeaders (6)BodyPre-request ScriptTestsSettingsCookies

Query Params

<input checked="" type="checkbox"/> Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/> question	چگونه فعال كنم؟			
Key	Value	Description		

BodyCookiesHeaders (5)Test Results

Status: 200 OKTime: 5.12 sSize: 1.46 KBSave as example

PrettyRawPreviewVisualizeJSON

```
1 {
2   "result": {
3     "final response": "در حساب کاربري خود وارد شويد و مراحل فعال سازي را انجام دهيد CDN به منوي",
4     "prompt": "Start of Conversation History  User : فعال كنم؟ CDN آيا ميتونم براي سايتم , chatbot: ميتوانيد براي سايت خود از سرويس . User
5       : استفاده كنيد CDN بله، ميتوانيد براي سايت خود از سرويس .در حساب کاربري خود وارد شويد و مراحل فعال سازي را انجام دهيد CDN به منوي , چگونه فعال كنم؟
6       .End of Conversation History"
7   }
}
```

[illegible]

نحوه‌ی استفاده از پروژه

تمامی سرویس های پروژه به صورت داکرایز شده در فایل `docker-compose` وجود دارد. به منظور استفاده از برنامه تنها کافی است فایل `docker compose` ران شود و سپس `table` مورد نظر مطابق با فیلدهای مذکور ساخته شود و سپس `api`های مربوطه فراخوانی شود. `Endpoint` هر یک از `api` ها نیز در تصاویر مربوط به هر یک مشخص است.