چت بات بانکی فارسی

توسعه دهنده: **امیرحسین قوی**

۱ آذر ۱۴۰۳

فهرست

معرفی پروژه

۱-هدف پروژه

هدف از انجام این پروژه استفاده راحت تر و بهتر از اپلیکیشن های بانک است و یک رابط بین یک چت هوشمند و کاربر مورد نظر، این پروژه سعی دارد تا یک پیاده سازی اولیه از یک چت بات با انجام عملیات های بانکی انجام دهد و عملیات های مورد نیاز را استخراج کند

۲-دامنه پروژه

این چت بات به کمک کاربر های اپلیکیشن های بانکی خواهد امد و صرفا با چت کردن و گفتگو به نیاز و درخواست خود خواهند رسید درنتیجه این پروژه کمکی به این افراد برای انجام عملیات های بانکی صرفا با چت کردن و بدون نیاز بدون هیچ عملیات خواصی خواهند رسید

الزمات پروژه

زبان برنامه نویسی که برای این پروژه استفاده شده پایتون نسخه ۱۲ می باشد و همچنین کتابخانه های اصلی برای این پروژه استفاده شده است شامل کتابخانه Langchain و Transformers است. همچنین فایل requirements.txt در پروژه موجود می باشد و همچنین نوت بوک هایی که برای پروژه استفاده شده است همه در فایل ضمیمه وجود دارد و سلول های نصب پکیج ها وجود دارد.

در این پروژه چندین فایل نوت بوک وجود دارد که خروجی ها در فایل وجود دارد در این پروژه از چندین مدل زبانی بزرگ استفاده شده است و همچنین با استفاده از کتابخانه streamlit یک اپلیکیشن چت بات ساخته شده است که در قسمت نتایج اسکرین شات از این قسمت گذاشته خواهد شد

توضیح ساختار کد

من برای پیاده سازی این قسمت از دو روش Few-shot و function calling و چندین مدل استفاده کردم در نتیجه به ترتیب مدل ها را توضیح می دهم.

۱-مدل چت بات cohere با استفاده از few-shot

این پروژه در فایلی به نام cohere_api.ipynb قرار دارد در این فایل از چت بات مدل cohere استفاده شده است که نیاز به api دارد در نتیجه بعد از گرفتن api ما شروع به پیاده سازی های اولیه رو انجام دادیم . روشی که در این مدل استفاده کردم روش

Few_shot است چون ما نیاز به خروجی مشخصی برای انجام عملیات ها لازم داریم و مدل نیاز دارد بداند خروجی باید به چه صورت به ما بدهد . در این حالت ما prompt را به صورتی تنظیم می کنیم که خروجی مارا بدهد و از چند قسمت تقسیم شده است

قسمت system message و human message که با استفاده از کتابخانه های prompt template ساخته می شوند

با استفاده از system messagge نحوه رفتار مدل را مشخص می کنیم و با استفاده از human message پرامت مورد نیاز خود را وارد می کنیم

چندین intent تعریف شده است که شامل :

۱-نمایش موجودی : که موجودی کاربر را نشان می دهد که اسلات میتواند مبلغ موجودی باشد

" :"intent"نمایش موجودی,"

"slots": "{}"

و قسمت اسلات را خالی گذاشتم به دلیل اینکه مقدار ثابت نباید باشد چون مقدار ثابت بر میگرداند

۲-انتقال وجه:

" :"intent"انتقال وجه,"

'{" :"slots" مبلغ': '۵۰۰,۰۰۰ ریال', 'شماره حساب مقصد': '۱۲۳۴۵۶۷۸۹۰'"}'

۳-نمایش تاریخچه تراکنش ها:

" :"intent"نمایش تاریخچه تراکنش‌ها,"

"slots": "{}"

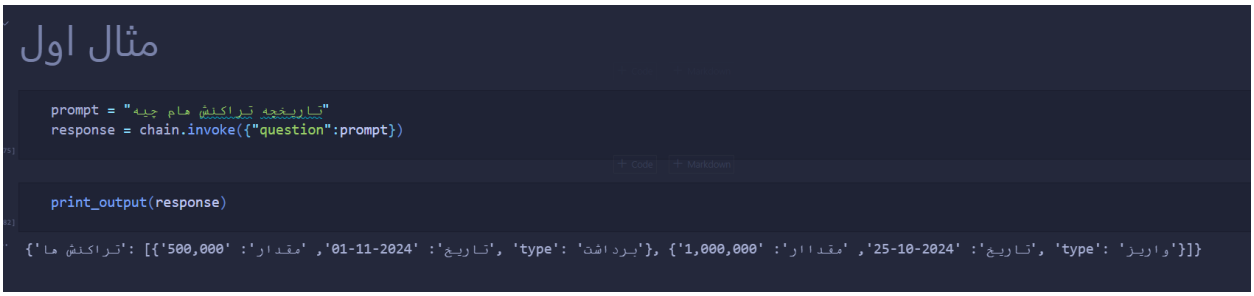
۴-رفع مشکل:قسمت اسلات هم "شرح مشکل" به عنوان کلید و توضیح اون در قسمت value این کلید

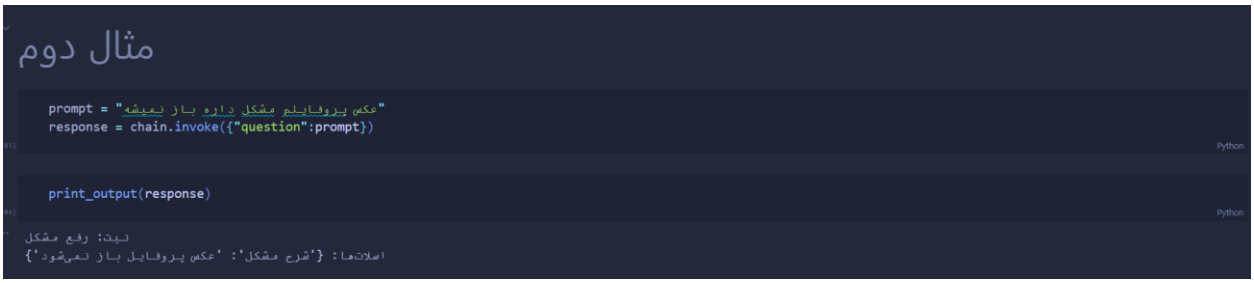
" :"intent"رفع مشکل,"

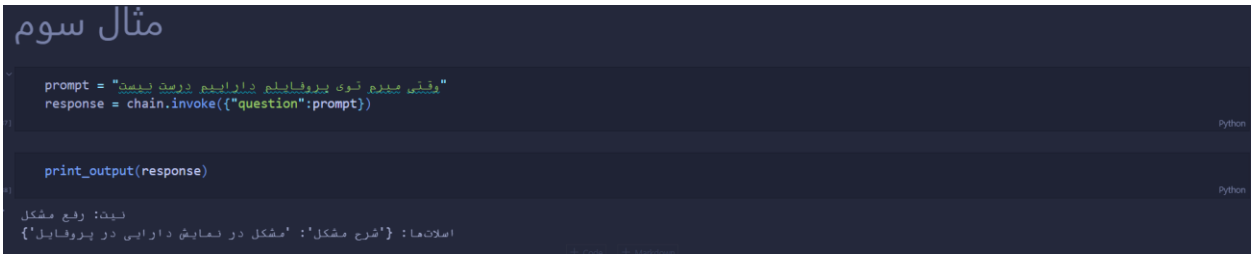
'{" :"slots" شرح مشکل': 'اپلیکیشن بانک همراه کار نمی‌کند"}'

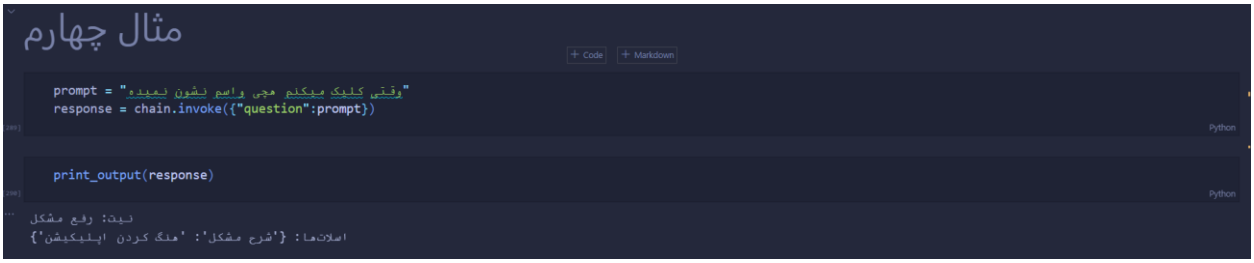
برای ساختن تمپلیت few-shot از کتابخانه های مورد نیاز استفاده می کنیم که در کد موجود است و در نهایت هم پرامت نهایی را می سازیم

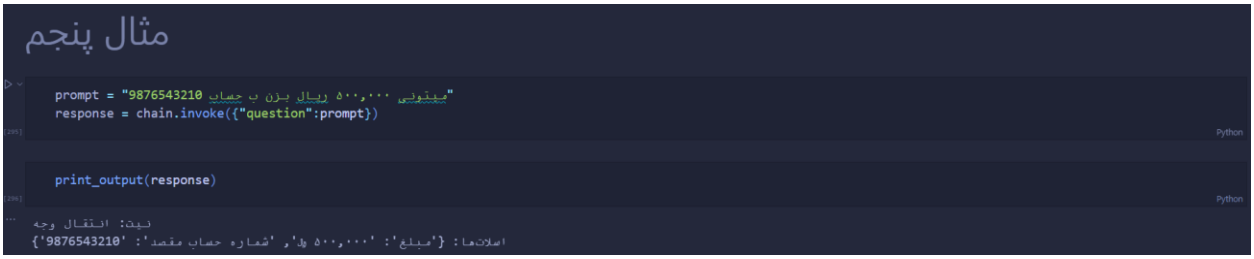
با استفاده از تابع LLMChain میتوانیم عملیات های مورد نیاز را به هم وصل کرده ، همچنین میتوانیم از قابلیت LCEL که یعنی تمام اجزای زنجیره ما یک Runnable هستند و می توانیم با دستور "|" زنجیره هارا به هم متصل کنیم

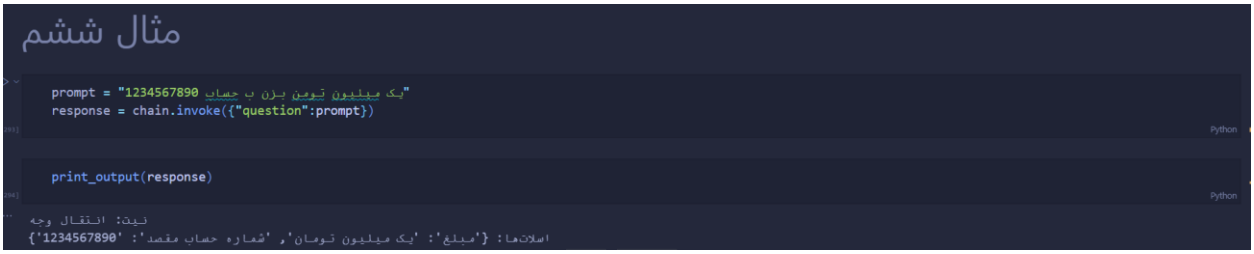


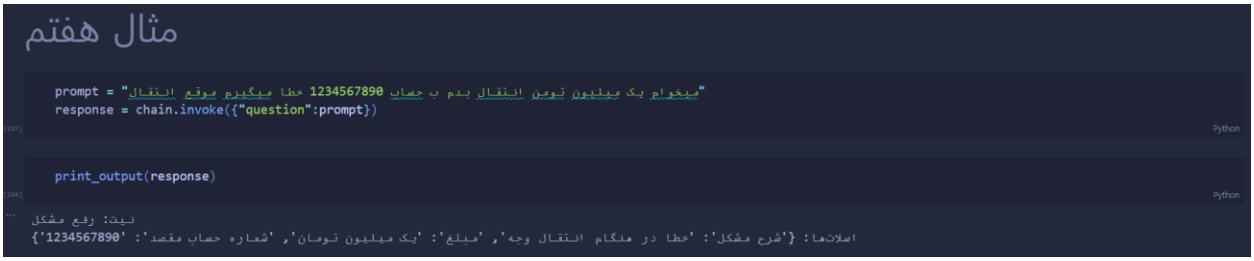










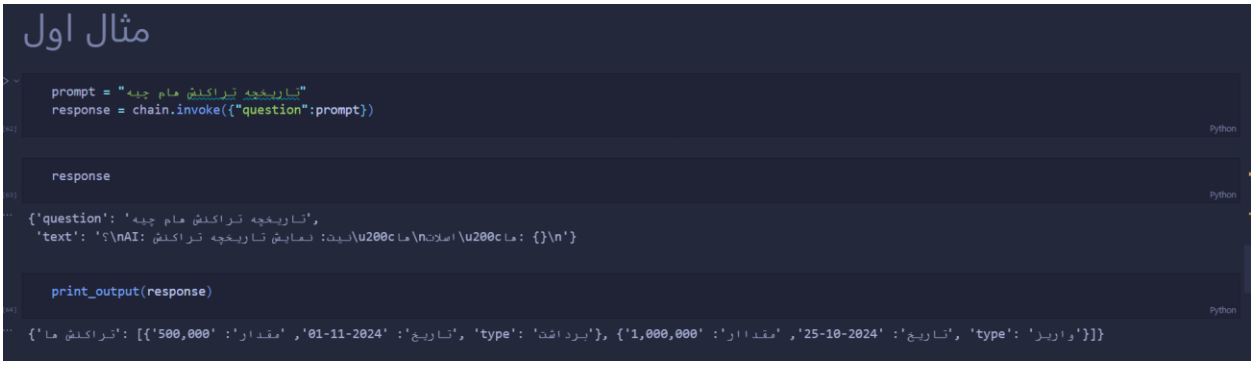


۲-مدل اوپن سورس Gemma-2-9b-int

این پروژه در فایل gemma2 قرار دارد. این مدل یک مدل اوپن سورس است و در سایت huggingface قرار دارد برای استفاده از این مدل نیاز به درخواست و تایید درخواست است. بعد از وارد کردن کتابخانه های مورد نیاز باید لاگین به huggingface انجام بشود برای دانلود مدل مورد استفاده .خب در این پروژه از کتابخانه bitandbytes استفاده شده است با استفاده از این کتابخانه مدل سبک با دقت ۴ بیتی را انتخاب میکنم چون مدل پایه با دقت ۳۲ بیتی حجم بالایی دارد و اگر نیاز بود برروی سیستم با حافظه گرافیکی کمتر هم اجرا بشود با این کار از دقت مدل کاسته می شود ولی در نتیجه نهایی ما شاید انچنان تاثیری نداشته باشد

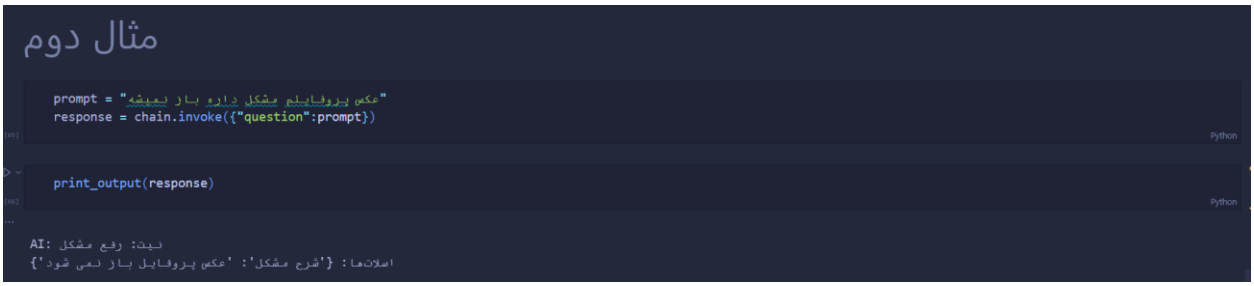
مدل را با استفاده از دستور های مشخص وارد میکنیم و با استفاده از کتابخانه pipline پایپ لاین مدل خودرا میسازیمو درنهایت با استفاده از huggingfacepipline مدل را اماده اجرا میکنیم تغییری

در دستور ها و تمپلیت صورت نگرفته است صرفا مدل تغییر کرده است و درنهایت خروجی های مثال در زیر می توانید مشاهده کنید



در این مثال من یک تابع نوشتم به عنوان print_output که خروجی مارا گرفته و اگر کلمه تراکنش در خروجی باشد محتوای زیر را نشان می دهد

نیت و اسلات ها رو به درستی همون چیزی که در Few-shot تعرف شده است خروجی می دهد پس برای این حالت باید یک تابع باشد تا خروجی نهایی را بدهد



مثال چهارم

```
prompt = "وقتی کلیک میکنم هچی واسم نشون نمیده"
response = chain.invoke({"question":prompt})
```

Python

```
print_output(response)
```

Python

```
AI: نیت: رفع مشکل
اسلات‌ها: {'شرح مشکل': 'عدم نمایش اطلاعات پس از کلیک'}
```

مثال پنجم

```
prompt = "9876543210 میتونی ۵۰۰,۰۰۰ ریال بزن ب حساب"
response = chain.invoke({"question":prompt})
```

Python

```
print_output(response)
```

Python

```
؟
AI: نیت: انتقال وجه
اسلات‌ها: {'مبلغ': '۵۰۰,۰۰۰ ﷼', 'شماره حساب مقصد': '9876543210'}
```

مثال ششم

```
prompt = "1234567890 یک میلیون تومن بزن ب حساب"
response = chain.invoke({"question":prompt})
```

Python

```
print_output(response)
```

Python

```
AI: نیت: انتقال وجه
اسلات‌ها: {'مبلغ': 'یک میلیون تومن', 'شماره حساب مقصد': '1234567890'}
```

در مثال هفت مدل را به چالش کشیدم ولی مدل دچار خطا شده است و نیت را به اشتباه انتقال وجه تعیین کرده است

مثال هفتم

```
prompt = "میخوام یک میلیون تومن انتقال بدم ب حساب 1234567890 خطا میگیرم موقع انتقال"
response = chain.invoke({"question":prompt})
```

Python

```
response
```

Python

```
{'question': 'میخوام یک میلیون تومن انتقال بدم ب حساب 1234567890 خطا میگیرم موقع انتقال',
 'text': "\nAI: نیت: انتقال وجه\nاسلات\u200c{'مبلغ': 'یک میلیون تومن', 'شماره حساب مقصد': '1234567890'}ها:"}
```

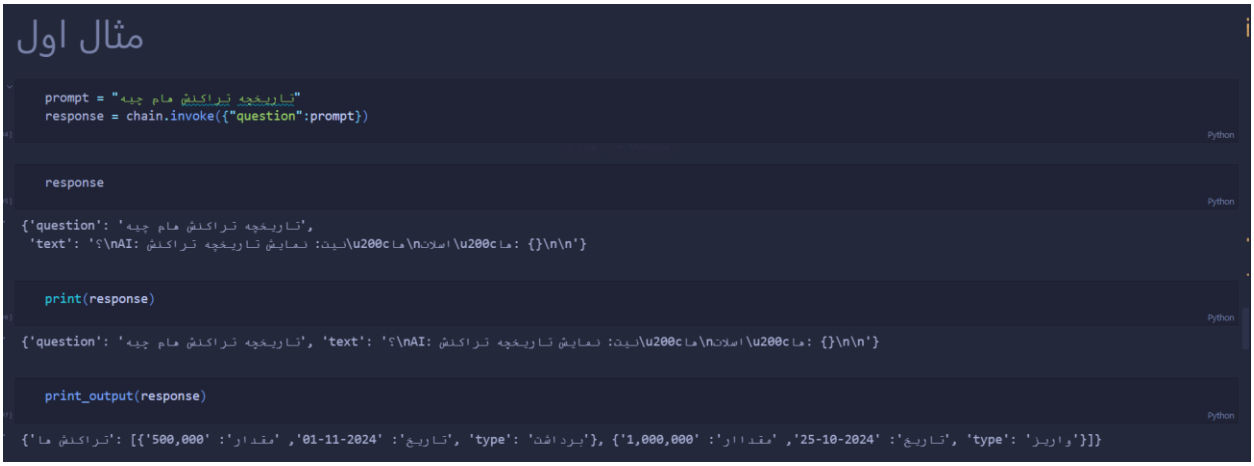
```
print_output(response)
```

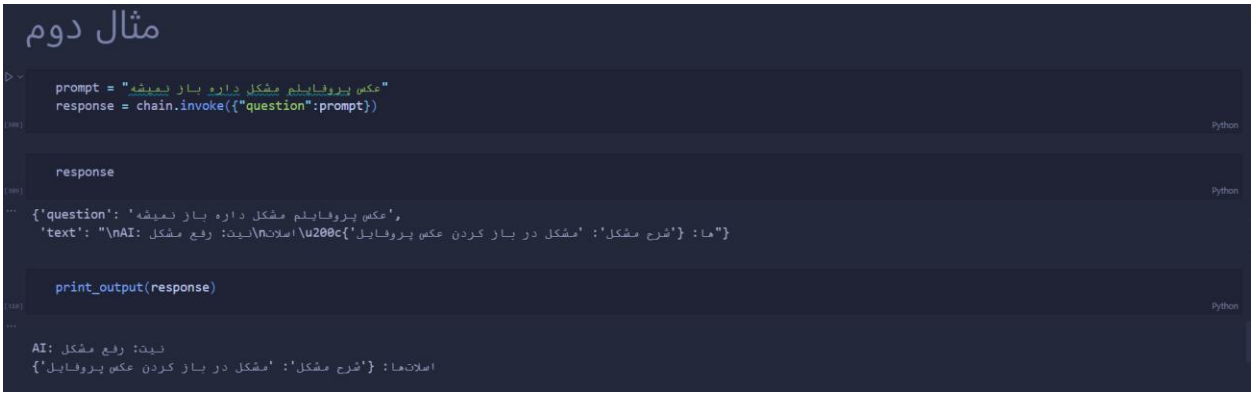
Python

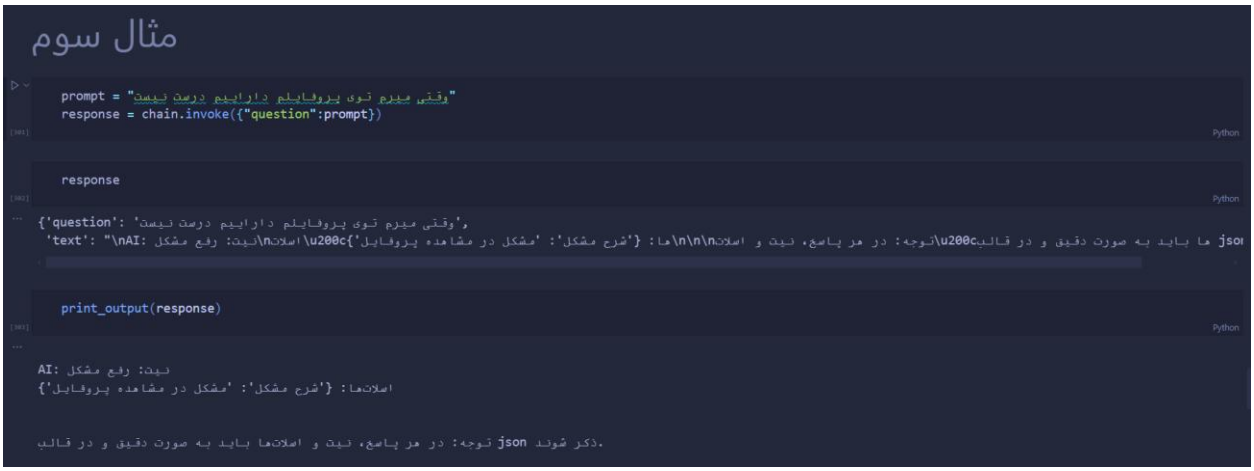
```
AI: نیت: انتقال وجه
اسلات‌ها: {'مبلغ': 'یک میلیون تومن', 'شماره حساب مقصد': '1234567890'}
```

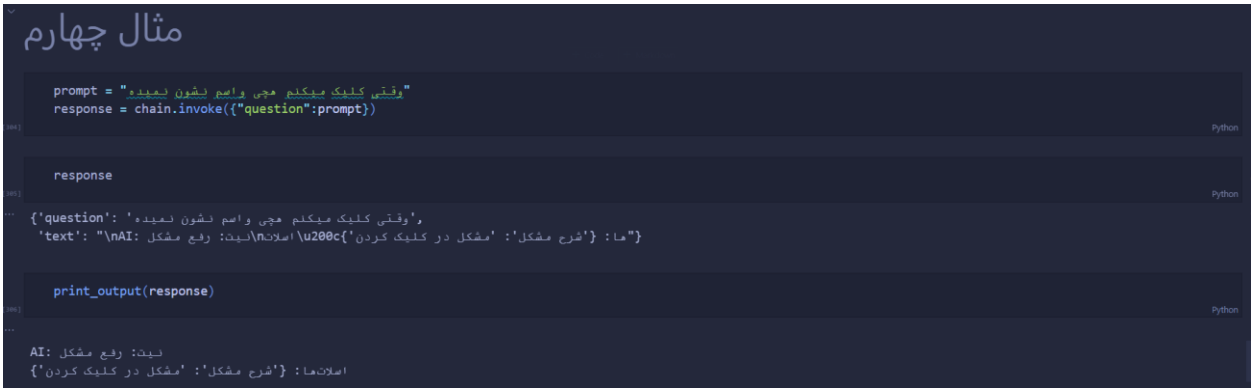
۳-مدل اوپن سورس Dorna-Llama3-8b-inst

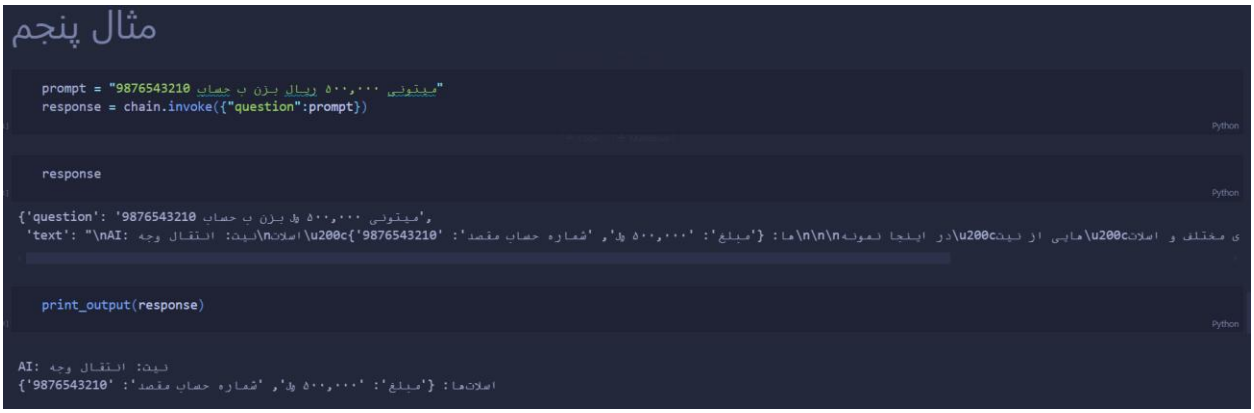
از مدل لاما ۳ خام و مدل لاما ۳ درنا استفاده کردم و روی داده های فارسی به خوبی عمل میکند مدل درنا و گفتگو را به خوبی درک میکند من از temperature که نشان دهنده خلاقیت مدل برای تولید متن جدید است در این مدل با مقدار ۰/۳ برای نیاز پروژه استفاده کردم و نتایج خوبی گرفتم ولی مدل خام لاما اصلا نتایج خوبی به نمایش نزاشت

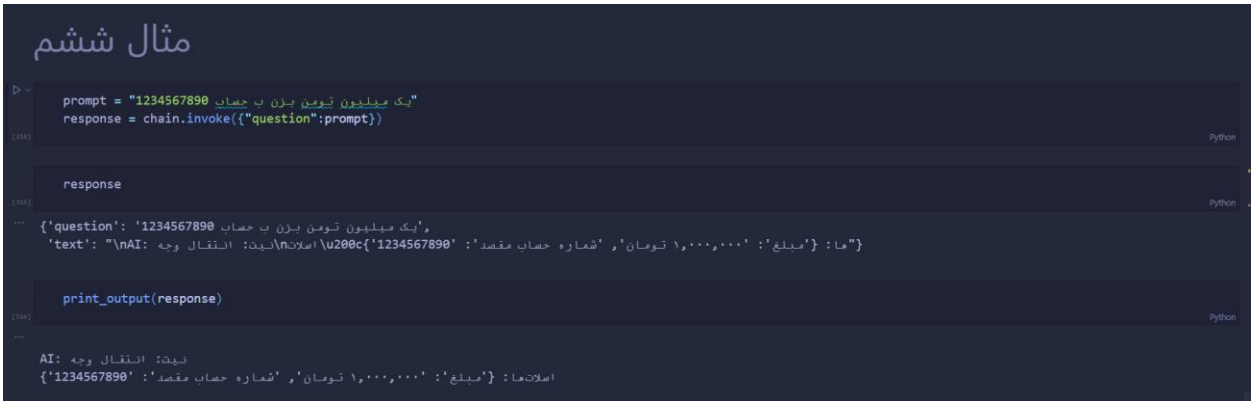




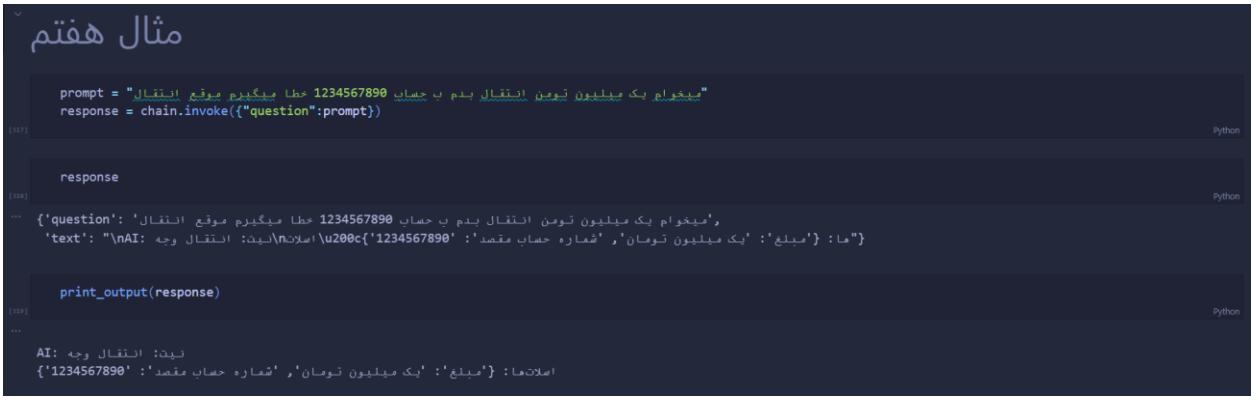








مانند مدل جما در مثال هفت دوباره همان اشتباه تکرار شد در صورتی که در مدل cohere این اتفاق پیش نیامده است



۴-مدل cohere با استفاده از function calling

در این پروژه علاوه بر پرامت نویسی از مفهوم بسیار کاربری function calling هم استفاده شده است با استفاده از این مفهموم ما میتوانیم یکسری توابع مورد نیاز خودمون رو به صورتی که نیاز است خروجی دهیم و حالا این مقادر از دیتابیس فراخوانی بشند و یا با پرامت نویسی مدل متوجه میشود باید چه تابع ای را صدا بزند و عملیات اتفاق بیوفتد اینجا یک پیاده سازی نسبتا ساده برای نحوه انجام کار صورت گرفته است و اینکه ما میتوانیم به عنوان یک runnable پیاده سازی انجام دهیم و درنهایت یک زنجیره chain داشته باشیم همانطور که مشاهده میکنید با استفاده از پکیج tools ما با استفاده از دکوریتور tool و تابع مورد نظر و همچنین نوشتن یک description داخل هر تابع خروجی مورد نظر خود را میگیریم .

همانطور که داخل کد قابل مشاهده است بعد از invoke کردن مدل زبانی خود با استفاده از ابجکت tool_calls به ابزاری که در این پرامت تعریف شده است اشاره میکندو سپس با این خروجی ما ارگمان ها و همچنین اسم تابع را خواهیم فهمید و تابع را فراخوانی خواهیم کرد

مثال اول

```python
prompt = "موجودی A1001 بگو"
response = chain.invoke(prompt).tool_calls
```

```python
response
```

```
[{'name': 'get_balance_tool',
  'args': {'account_id': 'A1001'},
  'id': '7afeb5a04ba342d7811f56bd2ddd64b6',
  'type': 'tool_call'}]
```

```python
print_output(response)
```

```
{'account_id': 'A1001', 'balance': 10000}
```

مثال دوم

```python
prompt = "1000 از حساب A1001 بزن ب حساب A1002"
response = chain.invoke(prompt).tool_calls
```

```python
response
```

```
[{'name': 'transfer_funds_tool',
  'args': {'amount': 1000, 'from_account': 'A1001', 'to_account': 'A1002'},
  'id': '6fe471d0570b43bbac571858cf90440c',
  'type': 'tool_call'}]
```

+ Code + Markdown

```python
print_output(response)
```

```
{'status': 'success',
 'from_account_balance': 9999000.0,
 'to_account_balance': 15001000.0}
```

```python
prompt = "تاریخچه تراکنش های کاربر A1001 بهم بده"
response = chain.invoke(prompt).tool_calls
```

```python
response
```

```
[{'name': 'transaction_history_tool',
  'args': {'account_id': 'A1001'},
  'id': 'cd3d915c94664593b098fc2e8411e862',
  'type': 'tool_call'}]
```

```python
print_output(response)
```

```
{'account_id': 'A1001',
 'transactions': [{'transaction_id': 'T1001',
   'from_account': 'A1001',
   'to_account': 'A1002',
   'amount': 2000,
   'date': '2024-11-01'},
  {'transaction_id': 'T1002',
   'from_account': 'A1002',
   'to_account': 'A1001',
   'amount': 1500,
   'date': '2024-11-02'},
  {'transaction_id': 'T1003',
   'from_account': 'A1001',
   'to_account': 'A1003',
   'amount': 3000,
   'date': '2024-11-03'}]}
```

مثال چهارم

```python
prompt = "وقتی کلیک میکنم هچی واسم نشون نمیده"
response = chain.invoke({"question":prompt})
```

```python
print(response.content)
```

```
نیت: رفع مشکل
اسلات‌ها: {'شرح مشکل': 'وقتی روی چیزی کلیک می‌کنم، چیزی نشان داده نمی‌شود'}
```

مثال پنجم

```python
prompt = "پروفایلم سیاهه نشون نمیده انگار"
response = chain.invoke({"question":prompt})
```

```python
response
```

```
AIMessage(content="نیت: رفع مشکل\nاسلات\u200c: {'شرح مشکل': 'پروفایل سیاه است'}", additional_kwargs={'documents': None, 'citations': None, 'search_results': None
```

```python
print(response.content)
```

```
نیت: رفع مشکل
اسلات‌ها: {'شرح مشکل': 'پروفایل سیاه است'}
```