به نام خدا

گزارش پروژه

پیاده‌سازی تولید متن با استفاده از بازیابی (RAG) بر روی گزارش‌های سازمانی

درس پردازش زبان طبیعی

استاد:

دکتر مرضیه داوود آبادی

توسعه‌دهندگان:

کامیار مرادیان زه‌آب ۹۹۵۲۲۱۰۴

ارشیا حسین‌زاده ۹۹۵۲۱۲۰۸

یزدان ماستری فراهانی ۹۹۵۲۲۰۵۹

دانشگاه علم و صنعت ایران

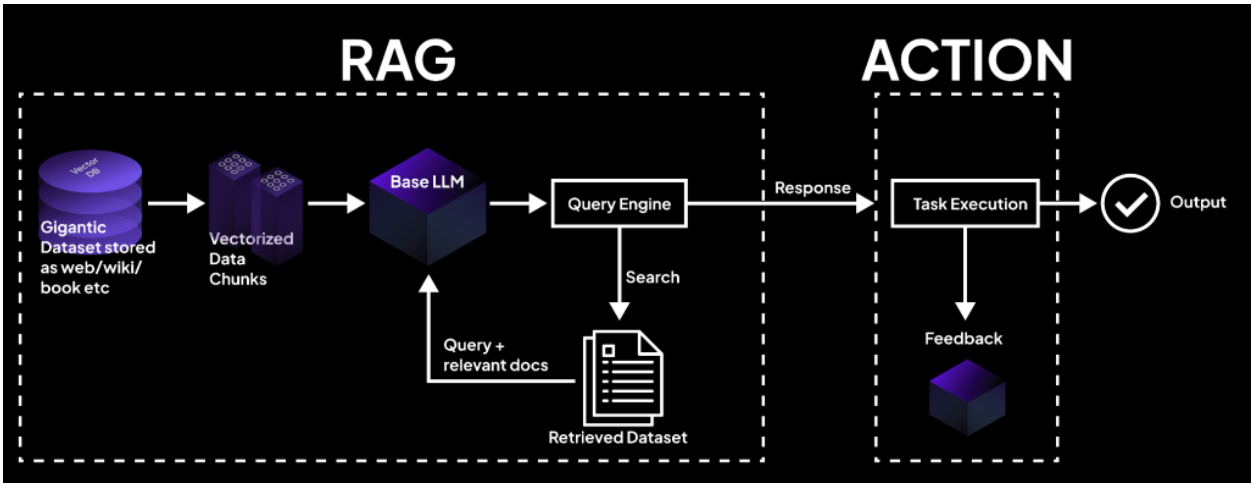
دانشکده مهندسی کامپیوتر

تیر ماه ۱۴۰۳

مقدمه

این پروژه با هدف پیاده‌سازی تولید متن با استفاده از تکنولوژی RAG بر روی گزارش‌های سازمانی است. RAG یک رویکرد پیشرفته در زمینه هوش مصنوعی است که با ترکیب مدل‌های زبانی بزرگ و تکنیک‌های بازیابی اطلاعات، توانایی تولید محتوای دقیق و به‌روز را دارد. اهمیت استفاده از RAG در پاسخ‌دهی به سوالات سازمانی و تحلیل گزارش‌های سازمانی در ارائه اطلاعات سریع و قابل‌اعتماد به مدیران و کارکنان سازمان نهفته است.

با بهره‌گیری از این تکنولوژی، می‌توان پاسخ‌های متنی با کیفیت بالا و اطلاعات مرتبط و به‌روز را به سرعت تولید و ارائه کرد، که این امر به تصمیم‌گیری‌های بهتر و کارآمدتر در سازمان کمک می‌کند. اهداف کلی پروژه شامل ارتقاء دقت و کارایی در تولید و تحلیل گزارش‌های سازمانی، بهبود پاسخ‌دهی به سوالات مرتبط با داده‌های سازمانی، و افزایش تجربه کاربری با ارائه اطلاعات دقیق و به‌موقع است. این پروژه می‌تواند تحولی اساسی در نحوه مدیریت و استفاده از اطلاعات سازمانی ایجاد کند.

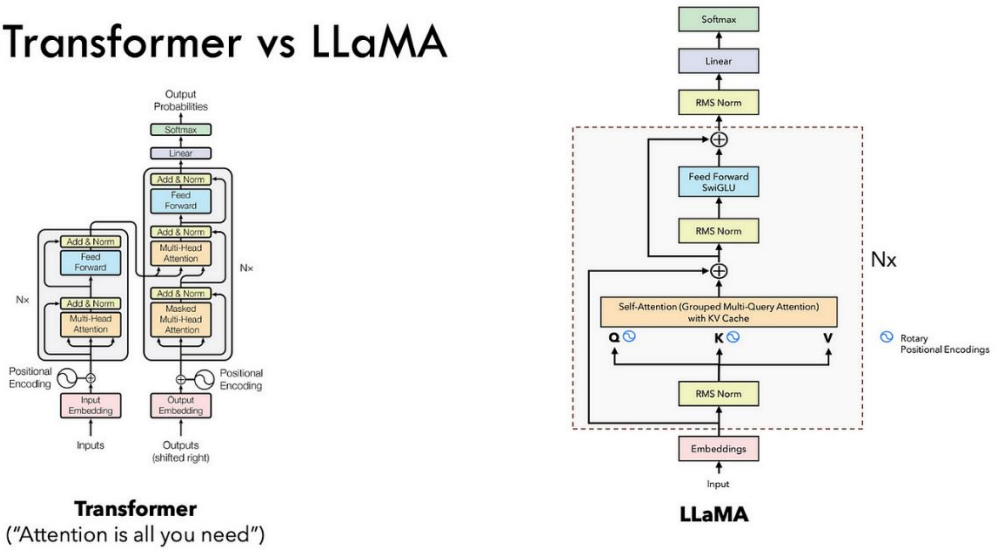


پیش‌زمینه تکنولوژی

مدل‌های LLaMA 3 و LaBSE از جمله پیشرفته‌ترین مدل‌های زبانی و بازیابی اطلاعات هستند که در حوزه‌های مختلف هوش مصنوعی و پردازش زبان طبیعی (NLP) کاربردهای گسترده‌ای دارند. این مدل‌ها با توانایی‌های منحصر به فرد خود در درک و تولید زبان انسانی، امکان ترکیب با سیستم‌های بازیابی اطلاعات را به صورت کارآمد فراهم می‌کنند.

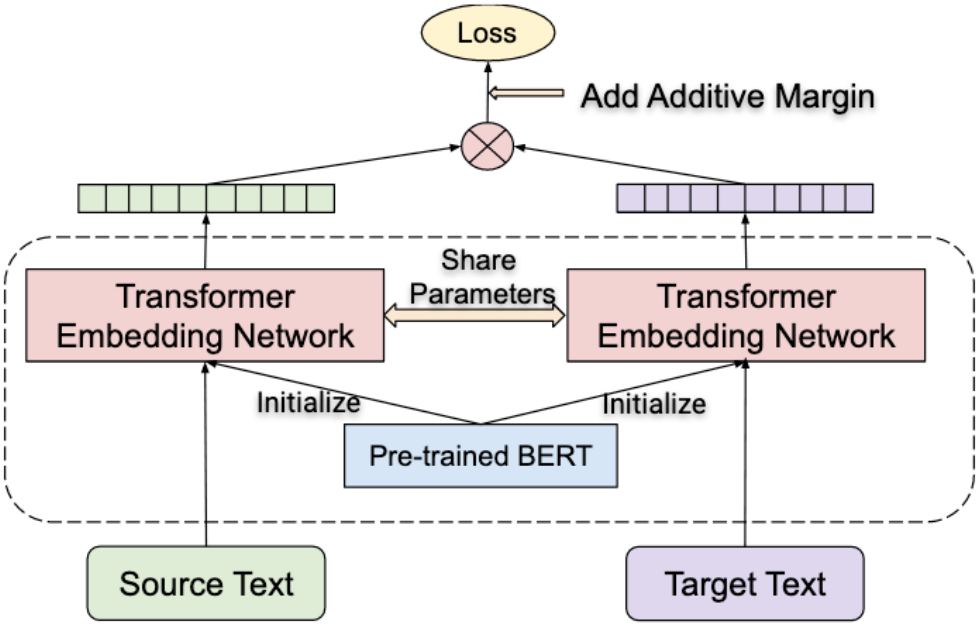
مدلLLaMA 3

LLaMA 3 یک مدل زبانی بزرگ است که توسط شرکت‌های پیشرو در هوش مصنوعی توسعه یافته است. این مدل با استفاده از میلیاردها داده آموزشی، توانایی تولید متن‌های طبیعی و متنوع را داراست LLaMA 3 .می‌تواند در زمینه‌های مختلفی مانند نوشتن مقالات، پاسخ‌دهی به سوالات پیچیده، و حتی تولید محتوای خلاقانه مورد استفاده قرار گیرد. یکی از ویژگی‌های برجسته این مدل، قابلیت آن در درک عمیق‌تر و تولید متون با زمینه‌های پیچیده و تخصصی است.



مدلLaBSE

LaBSE (Language-agnostic BERT Sentence Embedding) یک مدل مخصوص تولید بردارهای جملات است که به طور خاص برای کاربردهای چندزبانه طراحی شده است. این مدل قادر است جملات را در زبان‌های مختلف به فضای برداری منتقل کند، به گونه‌ای که جملات مشابه در زبان‌های مختلف بردارهای نزدیک به هم داشته باشند LaBSE . می‌تواند در سیستم‌های ترجمه، جستجوی معنایی و تطبیق محتوا در زبان‌های مختلف بسیار مفید باشد.



ترکیب با سیستم‌های بازیابی اطلاعات

ترکیب مدل‌های LLaMA 3 و LaBSE با سیستم‌های بازیابی اطلاعات می‌تواند کارایی و دقت این سیستم‌ها را به طور چشمگیری افزایش دهد. مدل LLaMA 3 با توانایی تولید محتوای پیچیده و طبیعی، می‌تواند به تولید پاسخ‌های متنی دقیق و جامع کمک کند. از سوی دیگر، مدل LaBSE با ارائه بردارهای جملات و قابلیت جستجوی معنایی، می‌تواند فرآیند بازیابی اطلاعات را بهینه‌سازی کرده و نتایج مرتبط‌تری ارائه دهد.

به عنوان مثال، در یک سیستم بازیابی اطلاعات برای پاسخ‌دهی به سوالات چندزبانه، می‌توان از LaBSE برای تطبیق سوالات و پاسخ‌ها در زبان‌های مختلف استفاده کرد و از LLaMA 3 برای تولید پاسخ‌های دقیق و مناسب به زبان کاربر بهره برد. این ترکیب می‌تواند منجر به افزایش دقت و رضایت کاربران از سیستم‌های بازیابی اطلاعات شود.

روند پیاده‌سازی

لود کردن داده

ابتدا هر پنج صفحه از داده های درون پی دی اف های input2 , input1 را خوانده و سپس با استفاده از نرمالایزر هضم آنها را نرمالایز کرده و درون فایل هایی به فرمت تکست ذخیره می‌کنیم. این فایل‌ها ورودی ایندکس llama هستند.

Retrival

در این مرحله llama index همان نقشه information retrival را ایفا می‌کند. به این صورت که مرتبط ترین بخش از داکیومنت را با توجه به کوئری که در ورودی برای آن تعریف می‌کنیم، بر می‌گرداند. (در اینجا سه بخش مرتبط را بر می‌گرداند)

Generator

سپس خروجی مرحله قبل با کوئری مربوط به آن به عنوان ورودی Generator (که در اینجا همان Llama3 است) به آن پاس داده می‌شود، سپس generator با توجه به آنها متن خروجی مورد نظر را تولید می‌کند.

Evaluation

در نهایت از مدل Alibaba-NLP/gte-large-en-v1.5 استفاده می‌کنیم تا امبدینگ خروجی تولید شده و پاسخ اصلی کوئری را بیابیم. سپس با استفاده از cosine-similarity، تشابه میان دو امبدینگ را محاسبه کرده و مجموع این امتیازات را بر تعداد کل کوئری‌ها محاسبه میکنیم، تا دقت مدل بدست‌آید.

نتایج و بررسی‌ها

در طول انجام پروژه، از پرامپت‌های متفاوتی استفاده کردیم تا به تولید بهترین پاسخ‌ها دست بیابیم. این تجربه به ما نشان داد که انتخاب صحیح پرامپت‌ها تأثیر بسزایی در دقت و کیفیت نتایج تولید شده دارد. دقت های بدست آمده از خروجی تولید شده توسط دو پرامپت متفاوت در عکس های زیر نشان داده شده است.

prompt = Write answer of questions just in persian

```
[33] # Add a new column for similarity scores
     merged_df['Similarity'] = 0.0

     # Compute the total score
     total_score = 0
     for index, row in tqdm(merged_df.iterrows(), total=merged_df.shape[0], desc="Calculating Similarities"):
         answer = row['Answer']
         model_answer = row['Model Answer']
         similarity = calculate_similarity(answer, model_answer)
         merged_df.at[index, 'Similarity'] = similarity
         total_score += similarity

     print(f"\nTotal similarity score: {total_score/len(merged_df)}")

     # Save the updated DataFrame to a new CSV file
     merged_df.to_csv('final_result.csv', index=False)

     Calculating Similarities: 100%|██████████| 139/139 [00:32<00:00,  4.30it/s]
     Total similarity score: 0.6392572406384585
```

prompt = You are a Persian chatbot capable of answering municipality questions. Answer all questions just in Persian.

```
[ ]  # Add a new column for similarity scores
     merged_df['Similarity'] = 0.0

     # Compute the total score
     total_score = 0
     for index, row in tqdm(merged_df.iterrows(), total=merged_df.shape[0], desc="Calculating Similarities"):
         answer = row['Answer']
         model_answer = row['Model Answer']
         similarity = calculate_similarity(answer, model_answer)
         merged_df.at[index, 'Similarity'] = similarity
         total_score += similarity

     print(f"\nTotal similarity score: {total_score/len(merged_df)}")

     # Save the updated DataFrame to a new CSV file
     merged_df.to_csv('final_result.csv', index=False)

     Calculating Similarities: 100%|██████████| 139/139 [00:27<00:00,  4.98it/s]
     Total similarity score: 0.6586930488082144
```

موانع و چالش‌ها

از موانع و چالش‌ها می‌توان به موارد زیر اشاره کرد که در ادامه به نحوه پاسخ‌دهی به هرکدام نیز اشاره کوتاهی شده است:

1. تغییر فرمت داده‌ها

در این پروژه، اولین مشکلی که با آن مواجه شدیم، خواندن فایل سوالات بود. این فایل به فرمت ورد (Word) قرار داشت و مدیریت و استخراج داده‌ها از آن دشوار بود. برای اصلاح این مشکل، ابتدا تصمیم گرفتیم فایل ورد را به فرمت اکسل (Excel) تغییر دهیم. فرمت اکسل به دلیل ساختار جدولی و سازماندهی بهتر داده‌ها، امکان پردازش و تحلیل آسان‌تری را فراهم می‌کند .

2. اتمام منابع GPU , Memory

برای حل این مشکل نحوه طراحی مدل را با استفاده از پارامتر های خود مدل تغییر دادیم تا هنگام لود کردن آن از منابع کمتری استفاده کند. (به سه پارامتر آخر توجه شود)

```
# Load the model with specific parameters
model = AutoModelForCausalLM.from_pretrained(
    "NousResearch/Meta-Llama-3-8B-Instruct",
    device_map="cuda",
    torch_dtype=torch.float16,
    low_cpu_mem_usage=True,
    trust_remote_code=True,
    load_in_8bit=True
)
```

3. سرعت پایین در تولید متن

مقداردهی برخی از پارامتر های مدل که تاثیر مثبتی در کانفیگ متن خروجی می‌گذارند و باعث افزایش سرعت و کیفیت آن می‌شوند.

علاوه بر این مورد، سوالات را به چندین بخش تقسیم کرده و هر بخش را در اکانت های مختلفی درون google colab اجرا کردیم تا از اتلاف وقت جلوگیری کنیم.

```
# Initialize HuggingFace LLM with the preloaded model and tokenizer
llm = HuggingFaceLLM(
    model=model,
    tokenizer=tokenizer,
    generate_kwargs={
        "do_sample": False,
        "temperature": 0.6,
        "top_p": 0.9,
    },
```

4. افزایش کیفیت متن خروجی

متن خروجی ابتدایی تولید شده توسط مدل بسیار ضعیف و شامل بخش های انگلیسی متعددی بود. برای بهتر کردن خروجی مدل، شروع به استفاده از پرامپت های متفاوتی کردیم تا به بهترین نتیجه ممکن برسیم. این مورد تاثیر بسزایی در افزایش کیفیت متن خروجی داشت.

علاوه بر پیدا کردن پرامپت مناسب، توکن های ورودی را با توجه به تعداد توکن های دریافتی توسط مدل llama (8192 توکن) محدود کردیم، تا فایل های ورودی به درستی توسط مدل خوانده شوند.

آموخته‌ها

تجربه کار با مدل‌های LLaMA 3 و LaBSE نشان داد که این مدل‌ها چقدر در تولید محتوای باکیفیت و بازیابی اطلاعات دقیق مؤثر هستند. به‌خصوص، LLaMA 3 توانایی تولید متون طبیعی و LaBSE در مدیریت تطبیق معنایی جملات در زبان‌های مختلف بسیار مفید بود.

ترکیب مدل‌های زبانی پیشرفته با سیستم‌های بازیابی اطلاعات می‌تواند دقت و کیفیت نتایج را به طور چشمگیری افزایش دهد. این ترکیب نه تنها باعث بهبود در تولید پاسخ‌های دقیق شد، بلکه توانست زمان پاسخ‌دهی را نیز بهینه کند.

پیشنهادات

پیشنهادات برای انجام این پروژه بسیار است که در بخش چالش‌ها و موانع به بررسی بخشی از آنها پرداختیم که به صورت خلاصه می‌توان به موارد زیر اشاره کرد:

- استفاده از تکنیک‌های مدیریت GPU و یا استفاده از kaggle
- مقداردهی صحیح پارامتر های مدل در جهت تولید متن خروجی استاندارد
- استفاده از مولد های سنگین جهت گرفتن پاسخ های مناسب و طبیعی نسبت به کوئری‌ها
- نرمال‌سازی داده اولیه قبل از استفاده

منابع

1. https://medium.com/@tejaswi_kashyap/rag-processing-using-llamaindex-43d9786f9d8e
2. https://arxiv.org/abs/2005.11401
3. https://www.freecodecamp.org/news/mastering-rag-from-scratch/
4. https://www.smashingmagazine.com/2024/01/guide-retrieval-augmented-generation-language-models/
5. https://learnbybuilding.ai/tutorials/rag-from-scratch
6. https://huggingface.co/NousResearch/Meta-Llama-3-8B-Instruct
7. https://chatgpt.com