Retrieval-Augmented Generation

Gilyoung Cheong, Qidu Fu, Junichi Koganemaru, Xinyuan Lai, Sixuan Lou, Dapeng Shang

April 26, 2024

Introduction

Introduction

- Modern generative AI systems (ChatGPT, Google Gemini, etc.) require quality data to be fed to them in order for their responses to be useful.
- In many cases, when a user provides a query to the generative AI, the system utilizes a form of Retrieval-Augmented Generation (RAG) to rank data and feed them into a Large Language Model (LLM). The most relevant data is then summarized by the LLM and ultimately reported back to the user.

Objectives

The main goal of this project is to build a RAG model designed to sift through large pieces of data and rank the data based on their relevancy to a given prompt. There are two main problems that we want to address.

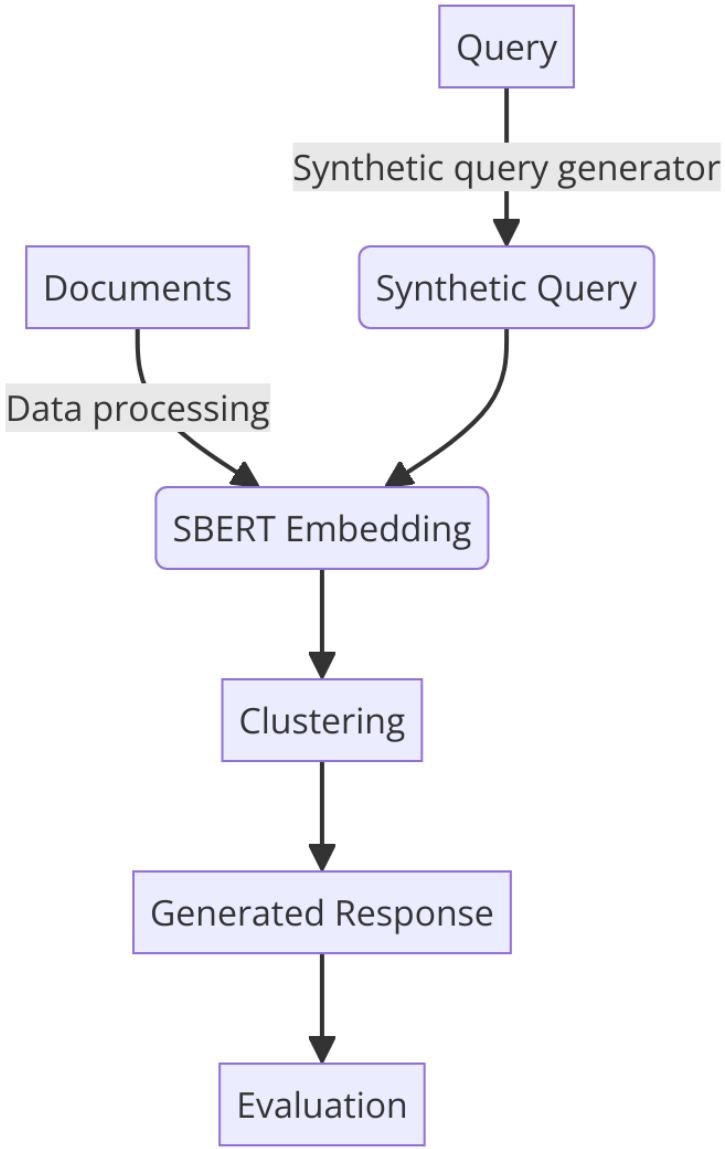
- Time efficiency: since generative AI systems rely on retrieving information from massive amounts of data, we aim to build a model that can rank and retrieve relevant queries quickly.
- Identifying metric and algorithm for ranking relevancy: in order to rank the prompts based on their relevancy to the original query, we aim to find an efficient algorithm and a ``good'' metric in order for us to assess the qualitative performance of our model. Our goal is to identify such a metric so that the response queries can be thought of as ``useful'' from a user perspective.

Dataset

Aware provided us with a dataset containing 5 million public Reddit submissions in the form of a .parquet file (552.1MB), containing important information including the text body, the subreddit the text is pulled from, etc.

aware_post_type	aware_created_ts	reddit_id	reddit_name	reddit_created_utc	reddit_author	reddit_text
submission	2023-04-02T13:58:03	129sqka	t3_129sqka	1,680,458,283	MoodyStarGirl	That's it.
comment	2023-04-02T14:32:57	jeounwc	t1_jeounwc	1,680,460,377	Lost_Treat_6296	We should make the chai tea latte with the same standard w...
comment	2023-04-02T14:48:18	jeowus2	t1_jeowus2	1,680,461,298	MoodyStarGirl	Oh like using the chai tea bags?
comment	2023-04-02T14:48:49	jeowxe5	t1_jeowxe5	1,680,461,329	Lost_Treat_6296	No, the whole half water and half milk thing
comment	2023-04-02T21:59:22	jeqiuw3	t1_jeqiuw3	1,680,487,162	MoodyStarGirl	That's a lot of water :(
comment	2023-04-02T14:41:14	jeovux3	t1_jeovux3	1,680,460,874	ateez-lvr	big disagree here tbh. the water to chai to milk concentration...
comment	2023-04-02T14:47:45	jeowry9	t1_jeowry9	1,680,461,265	MoodyStarGirl	O.o they taste watered down with the water, at least to me, a...

Overview of model architecture



Overview of model architecture

Our baseline RAG model is based on the following.

- Vector embedding using SBERT: the comments in the reddit database are embedded into $\mathbb{R}^{1024}$ using a pre-trained **transformer**. The transformer consists of stacked encoders that derives semantic meaning from the sentences and represents this information as vectors in $\mathbb{R}^{1024}$. The sentences and their semantic meanings can later be compared using **cosine similarity**.
- Clustering: to speed up the runtime, we grouped the comments into clusters using **K-means clustering** based on their vector representations with respect to cosine similarity. We stored the **average vector representations** of each cluster for quick comparison. This allowed us to cut the runtime down to under 1 second on average.

Overview of model architecture

While our baseline RAG model retrieved comments quickly, the quality of the responses was not always optimal. To improve the performance of the baseline model we incorporated the following.

- Synthetic query generation: given a String *s* modeling a user's input query, we feed *s* into a **synthetic query generator** to generate similar queries. We then rank the synthetic queries using cosine similarity and store the ones most similar to the original query with respect to this metric.
- Ranking by average cosine similarity: We then compute the cosine similarity between each comment in the database to the stored queries, and use the average as a measure of relevancy to the original prompt. The comments are then ranked by the averaged cosine similarity.

Main findings

We found that clustering significantly improved the runtime of our model, cutting down runtime from 5 seconds without clustering to under 1 second with clustering.

```
response("How many PTOs does a regular employee have a year?")
✓ 5.3s

Query: How many PTOs does a regular employee have a year?
Responses
1: Not sure about PTO, but the most ppto you can earn in a year is 48 hours, with the exception of a couple of states which are unlimited by state law.
2: All associates earn PPTO, which is intended for emergencies (car won't start, you're sick, etcFull-time associates also earn PTO, which is intended f
3: Full timers generally earn pto faster. But most part timers actually earn ppto faster. For most associates in most states, ppto is 1 hour for every 3
4: Just went through onboarding (for Walmart distribution and my chart clearly has 1 PPTO/30 hrs worked regardless of tenure. Regular pto is closer to h
5: I get 21 mins of ppto earned per shift (10 hrs and 19 mins of pto ... some states have different rules/laws but so I earn 84 minutes of ppto a week a
6: It is like pto
7: Yeah I'm aware of the benefits and pto. I just started recently but I'm getting scheduled mostly 10-7
8: Most places I've seen typically provide 0 PTO day one.
9: Not everyone gets ppto at the same rate. You start earning from day one and it’s available to use after 90 days. If you’re part time and less than 3
10: overtime, bonuses, PTO
Time used: 5.345 seconds
```

```
response_with_cluster("How many PTOs does a regular employee have a year?")
✓ 0.7s

Query: How many PTOs does a regular employee have a year?
Responses
1: Not sure about PTO, but the most ppto you can earn in a year is 48 hours, with the exception of a couple of states which are unlimited by state law.
2: All associates earn PPTO, which is intended for emergencies (car won't start, you're sick, etcFull-time associates also earn PTO, which is intended f
3: Full timers generally earn pto faster. But most part timers actually earn ppto faster. For most associates in most states, ppto is 1 hour for every 3
4: Just went through onboarding (for Walmart distribution and my chart clearly has 1 PPTO/30 hrs worked regardless of tenure. Regular pto is closer to h
5: I get 21 mins of ppto earned per shift (10 hrs and 19 mins of pto ... some states have different rules/laws but so I earn 84 minutes of ppto a week a
6: It is like pto
7: Yeah I'm aware of the benefits and pto. I just started recently but I'm getting scheduled mostly 10-7
8: Most places I've seen typically provide 0 PTO day one.
9: Not everyone gets ppto at the same rate. You start earning from day one and it’s available to use after 90 days. If you’re part time and less than 3
10: overtime, bonuses, PTO
Time used: 0.704 seconds
```

Main findings

We also found that **synthetic query generation and averaging** improved the quality of the retrieved comments, both from a human point of view and from two evaluation metrics that we designed to evaluate the quality of the comments.

	reddit_text	reddit_subreddit	textembedding	Cluster ID	cos_angles	avg_cos_angles
5678	Not sure about PTO, but the most ppto you can ...	WalmartEmployees	[-0.017012763768434525, 0.001875273184850812, ...	1	0.881989	0.919409
5655	Not everyone gets ppto at the same rate. You s...	WalmartEmployees	[-0.008390921168029308, 0.01311011053621769, -...	1	0.874638	0.902100
2620	I get 21 mins of ppto earned per shift (10 hrs...	WalmartEmployees	[0.003536654869094491, 0.007027565501630306, -...	1	0.877869	0.894189
1619	Full timers generally earn pto faster. But mos...	WalmartEmployees	[-0.008347814902663231, 0.010986099019646645, ...	1	0.879251	0.892648
80	1.00 = one hour.So you have almost an hour of ...	WalmartEmployees	[0.010056194849312305, 0.01743551343679428, -0...	1	0.863413	0.888485
301	All associates earn PPTO, which is intended fo...	WalmartEmployees	[-0.022059394046664238, -0.0017816615290939808...	1	0.879523	0.888055
3324	I was just remembering earlier that it's suppo...	WalmartEmployees	[-0.00040522910421714187, 0.007721611764281988...	1	0.857598	0.882631
2276	It is not a month if pto bro when you first ge...	BestBuyWorkers	[0.0026660107541829348, -0.008649304509162903,...	1	0.864908	0.881803
2572	Just as the title says... I'm at 184 hours PTO...	BestBuyWorkers	[0.01169597264379263, -0.0014767491957172751, ...	1	0.849545	0.876606
1517	It varies greatly by state/location and store ...	RiteAid	[0.0023047644644975662, 0.0036134349647909403,...	3	0.866713	0.876284

Data processing, embedding, retrieval and evaluation

Data selection

As a proof-of-concept, we chose 10 subreddits 'TalesFromYourBank', 'cabincrewcareers', 'Chase', 'KrakenSupport', 'WalmartEmployees', 'BestBuyWorkers', 'RiteAid', 'PaneraEmployees', 'FedEmployees' with $\sim$ 100k comments to test our baseline model and improved model.

Preprocessing

Prior to feeding the text data into SBERT for vectorization, we first took some steps to clean the data to improve the quality of the vectorization and retrieved comments. This included:

- Removing missing text data, including deleted and removed comments.
- Removing short and long words.
- Stemming and lemmatizing words using NLTK WordNetLemmatizer
- Removing emoticons and emojis
- Spelling out common chat abbreviations (e.g. afaik $\rightarrow$ As far as I know)

We deleted only 4000 comments (roughly 4%) in our 100k sample database.

Vectorization

Next, we fed the text data into **SBERT** for vectorization.

SBERT (Sentence-BERT) is a sentence transformer introduced by Nils Reimers and Iryna Gurevycvh in 2019 as a modification of the popular BERT (Bidirectional Encoder Representations from Transformers) model for computing word embeddings.

Comparing vector representations: cosine similarity

After computing the sentence embeddings using SBERT, we compare and rank the comments using cosine similarity. Given two non-zero embedded vectors $\boldsymbol{v}_1, \boldsymbol{v}_2 \in \mathbb{R}^{1024}$, the *cosine similarity* between $\boldsymbol{v}_1, \boldsymbol{v}_2$ is a number $S \in [-1, 1]$ defined via

$$S = \frac{\boldsymbol{v}_1 \cdot \boldsymbol{v}_2}{\|\boldsymbol{v}_1\| \, \|\boldsymbol{v}_2\|}, \tag{2.1}$$

where $\cdot$ denotes the standard Euclidean inner product and $\|\cdot\|$ is the standard Euclidean norm.

This is the metric on which SBERT is trained on.

Performance metrics

In order to evaluate our model, we compare our RAG model to a benchmark RAG model that ranks the comments using cosine similarity only to the original query. We propose the following evaluation metrics to measure the performance of our model and the baseline model.

Evaluation metric 1: cosine precision

Given a batch B of retrieved comments from a RAG model and two truth vectors $\boldsymbol{t}_1, \boldsymbol{t}_2 \in \mathbb{R}^{1024}$, we define the *cosine precision of* B to be a number p in $[-1, 1]$ defined via

$$p = \frac{1}{|B|} \sum_{\boldsymbol{v} \in B} \frac{\cos(\boldsymbol{t}_1, \boldsymbol{v}) + \cos(\boldsymbol{t}_2, \boldsymbol{v})}{2}.$$

This definition is motivated by the definition of *precision* or *positive predictive value*, which is a performance metric on a finite sample space measuring the relevant instance among the retrieved instances.

Evaluation metric 2: ranked cosine precision

We also propose a second evaluation metric that incorporates the ranking of the retrieved comments.

Let $B = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K\} \subseteq \mathbb{R}^{1024}$ be an ordered batch of retrieved comments, where the ordering is determined by cosine similarity and K is the number of retrieved comments. For every integer $1 \leqslant m \leqslant K$, we defined the *precision at rank m* to be the cosine precision of the truncated batch $B_m = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$, denoted by p_{B_m}. We then define the *ranked cosine precision of B* to be a number p_{rank} in $[-1, 1]$ defined via

$$p_{\text{rank}} = \frac{1}{|B|} \sum_{m=1}^{k} p_{B_m}. \tag{2.2}$$

In other words, the ranked cosine precision of B is the average of the cosine precisions of the truncated batches.

Main results and future work

Evaluation metric 1: cosine precision

Original query: How many PTOs does a regular employee have a year?

We fed the original query into an existing LLM to generate two ``ground truth'' responses.

Ground truth 1: Regular employees are entitled to 1 hour of paid time off per 30 hours worked, with a maximum of 48 hours per year.

Ground truth 2 = An employee is entitled to 68 hours of paid time off per year.

```
cos_precision(vectors_1, t_1, t_2)
```

```
array([[0.83458472]])
```

```
cos_precision(vectors_2, t_1, t_2)
```

```
array([[0.84573978]])
```

Evaluation metric 2: ranked cosine precision

Original query: How many PTOs does a regular employee have a year?

Ground truth 1: Regular employees are entitled to 1 hour of paid time off per 30 hours worked, with a maximum of 48 hours per year.

Ground truth 2 = An employee is entitled to 68 hours of paid time off per year.

```
cos_rank_precision(vectors_1
```

```
array([[0.8428778]])
```

```
cos_rank_precision(vectors_2
```

```
array([[0.85300408]])
```

Conclusion

- Through our work we found that **clustering** and comparing average embeddings can improve runtime significantly.
- We also found that **synthetic query generation and ranking via averaged cosine similarity** can meaningfully improve the performance of RAG models.

Future work

Testing performance of models on larger databases: our current model performs well on $\sim$ 100k comments, modifications are likely needed to ensure they scale well to larger databases.

a) Customizing cluster size: in our model we divided each subreddit into N clusters for a uniform N. To optimize runtime performance, larger subreddits will probably need to be grouped into more clusters. It would also be interesting to try other types of clustering methods.

b) Optimizing data management: using a vector database specifically built for vector embeddings (LanceDB, etc.) instead of using Pandas dataframes can likely improve runtime as well.

c) Building custom synthetic query generator, or use existing question in database.

d) Incoporating automatic RAG evaluation framework.

References

1. Chaudhary, Aditi, Karthik Raman, and Michael Bendersky. ``It's All Relative!--A Synthetic Query Generation Approach for Improving Zero-Shot Relevance Prediction." arXiv preprint arXiv:2311.07930 (2023).
2. Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).
3. Gao, Yunfan, et al. "Retrieval-augmented generation for large language models: A survey." arXiv preprint arXiv:2312.10997 (2023).
4. Reimers, Nils, and Iryna Gurevych. "Sentence-bert: Sentence embeddings using siamese bert-networks." arXiv preprint arXiv:1908.10084 (2019).

Thank you

Thank you for your attention!