

章节一：环境配置与模型下载

1. 环境配置

1 虚拟环境安装

(1) 本机

```
conda create -n flashrag python=3.11
conda activate flashrag
```

(2) 云平台

```
conda create -n flashrag python=3.11
conda init bash && source /root/.bashrc
conda activate flashrag
conda install ipykernel
ipython kernel install --user --name=flashrag # 设置kernel, --user表示当前用户, flashrag为虚拟环境
```

2 源码安装 FlashRAG

```
git clone https://github.com/RUC-NLPIR/FlashRAG.git
cd FlashRAG
pip install -e .
```

注意⚠️：可以直接用我提供的代码注解版

3 其它依赖包

Install vllm for faster speed

```
pip install vllm>=0.4.1
```

Install sentence-transformers

```
pip install sentence-transformers
```

Install pyserini for bm25

```
pip install pyserini
```

GPU(+CPU) version

```
conda install -c pytorch -c nvidia faiss-gpu=1.8.0
```

CPU-only version

```
conda install -c pytorch faiss-cpu=1.8.0
```

```
# 安装git和git-lfs【可选, 如果已安装, 则跳过】
sudo apt update
sudo apt install git
curl -s https://packagecloud.io/install/repositories/github/git-lfs/script.deb.sh | sudo bash
sudo apt-get install git-lfs
git lfs install
```

2. 下载模型

总共需要下载以下两个模型:

- E5-base-v2

```
git clone https://huggingface.co/intfloat/e5-base-v2
```

- Llama-3.2-3B-Instruct

```
git clone https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct
```

- bge-large-zh-v1.5

```
git clone https://huggingface.co/BAAI/bge-large-zh-v1.5
```

- Qwen2.5-1.5B-Instruct

```
git clone https://huggingface.co/Qwen/Qwen2.5-1.5B-Instruct
```

如果是中国用户, 推荐使用镜像平台 <https://hf-mirror.com/> 进行下载。

3. 下载数据集

数据集包含了query和对应的标准答案, 能够让我们评估构建的RAG系统的效果。

- **Demo数据集**: 为了简便, 我们从NQ中采样了17条数据作为toy dataset, 其地址在 [examples/quick_start/dataset/nq](#)。后续RAG流程将在这些questions上进行。

- **Demo文档集合**: 文档集合包含了大量的切分好的段落, 是RAG系统的外部知识来源。由于常用的文档集合往往非常大(~5G以上), 我们使用了一个通用知识数据集作为检索文档, 地址为 [examples/quick_start/indexes/general_knowledge.jsonl](#)。

由于文档数量较少, 可能很多query都无法搜到相关的文本, 这可能会影响最终的检索结果。

- **完整数据集**: 作者提供的仓库中同样提供了大量处理好的基准数据集, 如果需要获取完整的文

档集合, 可以访问我们 (https://huggingface.co/datasets/RUC-NLPIR/FlashRAG_datasets) 进行下载和使用。

```
git clone https://huggingface.co/datasets/RUC-NLPIR/FlashRAG_datasets
```

章节二：快速入门

Demo-1: 简单pipeline

我们在下面的步骤中会依次拆解各个步骤, 并展示相应的代码 (simple_pipeline.py)。

(1) 加载Config

首先我们需要加载 `Config`, 并填入之前下载的各种东西的路径。

`Config` 管理着实验中所有的路径以及超参数。在FlashRAG中, 各种参数可以通过 `yaml` 文件或者python内的字典传入`Config`中。

传入的参数会替换默认的内部参数, 具体的各种参数以及其默认值可以参考 [basic_config.yaml](#)。

在这里我们直接通过字典传入各种路径。

```
from flashrag.config import Config

config_dict = {
    'data_dir': 'dataset/',
    'index_path': 'indexes/e5_Flat.index',
    'corpus_path': 'indexes/general_knowledge.jsonl',
    'model2path': {'e5': <retriever_path>, 'llama2-7B-chat':
<generator_path>},
    'generator_model': 'llama2-7B-chat',
    'retrieval_method': 'e5',
    'metrics': ['em', 'f1', 'acc'],
    'retrieval_topk': 1,
    'save_intermediate_data': True
}

config = Config(config_dict=config_dict)
```

(2) 加载数据集以及Pipeline

接着, 我们需要加载相应的数据集和pipeline。

数据集可以通过前面设置的config自动读取, 我们只需要选择对应需要的test集合就可以了。

而pipeline的加载需要我们先根据想进行的RAG流程选择一个合适的pipeline。这里我们选择 `SequentialPipeline`，用于进行前面提到的Standard RAG流程。然后pipeline的过程中会自动加载对应的组件(检索器和生成器)，并完成各种初始化。

```
from flashrag.utils import get_dataset
from flashrag.pipeline import SequentialPipeline

all_split = get_dataset(config)
test_data = all_split['test']
pipeline = SequentialPipeline(config)
```

(3) 运行RAG流程

在完成上面的步骤之后，我们只需要调用 `pipeline` 的 `.run` 方法即可在数据集上运行RAG流程并生成评估结果。该方法返回一个包含中间结果和运行结果的数据集，其中 `pred` 属性的内容则是模型的预测结果。

需要注意的是，由于前面我们提供的是Toy文档集合和索引，因此结果可能会比较差。可以考虑换成自己的文档集合和索引，应该能够取得较好的结果。

在运行结束后，所有运行结果会被单独保存在本次实验对应的文件夹内，包括每个query的检索和生成结果，整体的评价分数等。

完整的代码如下：

```
from flashrag.config import Config
from flashrag.utils import get_dataset
from flashrag.pipeline import SequentialPipeline

config_dict = {
    'data_dir': 'dataset/',
    'index_path': 'indexes/e5_Flat.index',
    'corpus_path': 'indexes/general_knowledge.jsonl',
    'model2path': {'e5': <retriever_path>, 'Llama-3.2-1B-Instruct': <generator_path>},
    'generator_model': 'Llama-3.2-1B-Instruct',
    'retrieval_method': 'e5',
    'metrics': ['em', 'f1', 'acc'],
    'retrieval_topk': 1,
    'save_intermediate_data': True
}

config = Config(config_dict = config_dict)

all_split = get_dataset(config)
test_data = all_split['test']
pipeline = SequentialPipeline(config)

output_dataset = pipeline.run(test_data, do_eval=True)
print("---generation output---
```

```
print(output_dataset.pred)
```

运行命令:

```
cd FlashRAG/examples/quick_start

python simple_pipeline.py --model_path /root/autodl-tmp/models/Llama-3.2-1B-Instruct --retriever_path /root/autodl-tmp/models/e5-base-v2
```

```
○ (flashrag) root@autodl-container-48b1458428-957c1534:~/autodl-tmp/FlashRAG/examples/quick_start#
● (flashrag) root@autodl-container-48b1458428-957c1534:~/autodl-tmp/FlashRAG/examples/quick_start# python simple_pipeline.py --model_path /root/autodl-tmp/models/Llama-3.2-1B-Instruct --retriever_path /root/autodl-tmp/models/e5-base-v2
Retrieval process: 0% | 0/1 [00:00<7, 7it/s]
Use 'query': as retrieval instruction
Retrieval process: 100% | 1/1 [00:00-00:00, 4.85it/s]
Generation process: 0% | 0/5 [00:00<7, 7it/s]
Setting 'pad_token_id' to 'eos_token_id':None for open-end generation.
From v4.47 onwards, when a model cache is to be returned, 'generate' will return a 'Cache' instance instead by default (as opposed to the legacy tuple of tuples format). If you want to keep returning the legacy format, please set 'return_legacy_cache=True'.
Generation process: 20% | 1/5 [00:00-00:02, 1.86it/s]
Setting 'pad_token_id' to 'eos_token_id':None for open-end generation.
Generation process: 40% | 2/5 [00:00-00:01, 2.72it/s]
Setting 'pad_token_id' to 'eos_token_id':None for open-end generation.
Generation process: 60% | 3/5 [00:00-00:00, 3.72it/s]
Setting 'pad_token_id' to 'eos_token_id':None for open-end generation.
Generation process: 80% | 4/5 [00:01-00:00, 2.90it/s]
Setting 'pad_token_id' to 'eos_token_id':None for open-end generation.
Generation process: 100% | 5/5 [00:01-00:00, 3.40it/s]
---generation output---
{'em': 0.05882352941164705, 'f1': 0.10442498677792796, 'acc': 0.17647058823529413}
---generation output---
['C. V. Raman', 'The next Deadpool movie is Deadpool 3, release date is scheduled to be 2025', 'SG', 'South west', 'In the context of war, "HP" stands for "Health Points".', 'The Universal Declaration of Human Rights (UDHR)', 'Qatar', 'I cannot provide information about upcoming episodes of The Scandal TV show.', '2008', 'Dr. Sarojini Naidu', 'Adobe Flash Player 32', 'Pyotr Ilyich Tchaikovsky', 'There are 291 episodes in the original Dragon Ball Z anime series.', 'The cast of Law & Order: Special Victims Unit includes:\n\n1. Mariska Hargitay as Detective Olivia Benson\n2. Ice-T as Detective Od\n3. Edwin Hujar as Detective\n4. Variation\n5. Hawaii']
○ (flashrag) root@autodl-container-48b1458428-957c1534:~/autodl-tmp/FlashRAG/examples/quick_start#
```

Demo-2 : web端交互

备注: 一个小坑, 注意numpy版本适配问题, 要安装 1.26.4 版本。
pip install numpy==1.26.4

```
cd FlashRAG/examples/quick_start

cp ../methods/my_config.yaml .

streamlit run demo_en.py --server.port 6006 --server.address 0.0.0.0
```

输出结果:

```
○ (flashrag) root@autodl-container-48b1458428-957c1534:~/autodl-tmp/FlashRAG/examples/quick_start#
○ (flashrag) root@autodl-container-48b1458428-957c1534:~/autodl-tmp/FlashRAG/examples/quick_start# streamlit run demo_en.py --server.port 6006 --server.address 0.0.0.0

Collecting usage statistics. To deactivate, set browser.gatherUsageStats to false.

You can now view your Streamlit app in your browser.
URL: http://0.0.0.0:6006

INFO 11-30 12:04:31 config.py:350] This model supports multiple tasks: {'generate', 'embedding'}. Defaulting to 'generate'.
WARNING 11-30 12:04:31 arg_utils.py:1013] Chunked prefill is enabled by default for models with max_model_len > 32K. Currently, chunked prefill might not work with some features or models. If you encounter any issues, please disable chunked prefill by setting --enable-chunked-prefill=False.
INFO 11-30 12:04:31 config.py:1136] Chunked prefill is enabled with max_num_batched_tokens=512.
INFO 11-30 12:04:31 llm_engine.py:249] Initializing an LLM engine (v0.6.4.post1) with config: model='/root/autodl-tmp/models/Llama-3.2-1B-Instruct', speculative_config=None, tokenizer='/root/autodl-tmp/models/Llama-3.2-1B-Instruct', skip_tokenizer_init=False, tokenizer_mode=auto, revision=None, override_neuron_config=None, tokenizer_revision=None, trust_remote_code=False, dtype=torch.bfloat16, max_seq_len=131072, download_dir=None, load_format=LoadFormat.AUTO, tensor_parallel_size=1, pipeline_parallel_size=1, disable_custom_all_reduce=False, quantization=None, enforce_eager=False, kv_cache_dtype=auto, quantization_param_path=None, device_config=cuda, decoding_config=DecodingConfig(guided_decoding_backend='outlines'), observability_config=ObservabilityConfig(otlp_traces_endpoint=None, collect_model_forward_time=False, collect_model_execute_time=False), seed=0, served_model_names=['/root/autodl-tmp/models/Llama-3.2-1B-Instruct'], num_scheduler_steps=1, chunked_prefill_enabled=True, multi_step_stream_outputs=True, enable_prefix_caching=False, use_async_output_proc=True, use_cached_outputs=False, chat_template_text_format=string, mm_processor_kwargs=None, pooler_config=None)
INFO 11-30 12:04:32 model_runner.py:1072] Starting to load model /root/autodl-tmp/models/Llama-3.2-1B-Instruct...
Loading safetensors checkpoint shards: 0% Completed | 0/1 [00:00<, ?it/s]
Loading safetensors checkpoint shards: 100% Completed | 1/1 [00:00-00:00, 2.07it/s]
Loading safetensors checkpoint shards: 100% Completed | 1/1 [00:00-00:00, 2.07it/s]

INFO 11-30 12:04:33 model_runner.py:1077] Loading model weights took 2.3185 GB
INFO 11-30 12:04:33 worker.py:232] Memory profiling results: total_gpu_memory=23.68GiB initial_memory_usage=2.63GiB peak_torch_memory=3.49GiB memory_usage_post_profile=2.65GiB non_torch_memory=0.32GiB kv_cache_size=16.32GiB gpu_memory_utilization=0.85
INFO 11-30 12:04:33 gpu_executor.py:113] # GPU blocks: 33422, # CPU blocks: 8192
INFO 11-30 12:04:33 gpu_executor.py:117] Maximum concurrency for 131072 tokens per request: 4.08x
INFO 11-30 12:04:35 model_runner.py:1400] Capturing cudagraphs for decoding. This may lead to unexpected consequences if the model is not static. To run the model in eager mode, set 'enforce_eager=True' or use '--enforce-eager' in the CLI.
INFO 11-30 12:04:35 model_runner.py:1404] If out-of-memory error occurs during cudagraph capture, consider decreasing 'gpu_memory_utilization' or switching to eager mode. You can also reduce the 'max_num_seqs' as needed to decrease memory usage.
INFO 11-30 12:04:52 model_runner.py:1518] Graph capturing finished in 17 secs, took 0.15 GiB
2024-11-30 12:04:56.901 Examining the path of torch.classes raised: Tried to instantiate class '__path__._path', but it does not exist! Ensure that it is registered via torch::class.
Use 'query': as retrieval instruction
Processed prompts: 100% | 1/1 [00:00-00:00, 11.28it/s, est. speed input: 4300.13 toks/s, output: 157.98 toks/s]
Processed prompts: 100% | 1/1 [00:00-00:00, 4.60it/s, est. speed input: 340.70 toks/s, output: 207.16 toks/s]
```

⚡ FlashRAG Demo

This demo retrieves documents and generates responses based on user input.

Enter your prompt:

hello, how are you ?

Generate Responses

References

[1]: Hello

[2]: Construct a dialogue between two characters.?

[3]: Construct dialogue for two people having a pleasant exchange.?

[4]: Greet the visitor in a friendly way.?

[5]: Generate a conversation between two friends who met after a long time.?

Generated Responses:

Response with RAG:

I'm doing alright, just dealing with a lot of changes.

Response without RAG:

Hello! I'm doing great, thanks for asking. It's nice to connect with you. How can I assist you today? Do you have a specific question, need help with something, or just want to chat?

Demo-3: QA问答

chat_demo-1

```
cd FlashRAG/examples/quick_start
```

```
python chat_demo-1.py --model_path /root/autodl-tmp/models/Llama-3.2-1B-Instruct --retriever_path /root/autodl-tmp/models/e5-base-v2
```

输出结果:

```
⊙ (flashrag) root@autodl-container-48b1458428-957c1534:~/autodl-tmp/FlashRAG/examples/quick_start#
● (flashrag) root@autodl-container-48b1458428-957c1534:~/autodl-tmp/FlashRAG/examples/quick_start# python chat_demo-1.py --model_path /root/autodl-tmp/models/Llama-3.2-1B-Instruct --retriever_path /root/autodl-tmp/models/e5-base-v2
Load data from provided data
Retrieval process: 0%|          | 0/1 [00:00<?, 7it/s]
Use query: ' as retrieval instruction
Retrieval process: 100%|          | 1/1 [00:00<00:00, 3.97it/s]
Generation process: 0%|          | 0/1 [00:00<?, 7it/s]
Setting 'pad_token_id' to 'eos_token_id':None for open-end generation.
From v4.47 onwards, when a model cache is to be returned, 'generate' will return a 'Cache' instance instead by default (as opposed to the legacy tuple of tuples format). If you want to keep returning the legacy format, please set 'return_legacy_cache=True'.
Generation process: 100%|          | 1/1 [00:00<00:00, 3.33it/s]
('Joe Biden')
⊙ (flashrag) root@autodl-container-48b1458428-957c1534:~/autodl-tmp/FlashRAG/examples/quick_start#
```

Chat_demo-2

```
cd FlashRAG/examples/quick_start

python chat_demo-2.py --model_path /root/autodl-tmp/models/Llama-3.2-1B-Instruct --retriever_path /root/autodl-tmp/models/e5-base-v2
```

输出结果:

```
⊙ (flashrag) root@autodl-container-48b1458428-957c1534:~/autodl-tmp/FlashRAG/examples/quick_start#
● (flashrag) root@autodl-container-48b1458428-957c1534:~/autodl-tmp/FlashRAG/examples/quick_start# python chat_demo-2.py --model_path /root/autodl-tmp/models/Llama-3.2-1B-Instruct --retriever_path /root/autodl-tmp/models/e5-base-v2
Load data from provided data
Retrieval process: 0%|          | 0/1 [00:00<?, 7it/s]
Use query: ' as retrieval instruction
Retrieval process: 100%|          | 1/1 [00:01<00:00, 1.02s/it]
Generation process: 0%|          | 0/1 [00:00<?, 7it/s]
Setting 'pad_token_id' to 'eos_token_id':None for open-end generation.
From v4.47 onwards, when a model cache is to be returned, 'generate' will return a 'Cache' instance instead by default (as opposed to the legacy tuple of tuples format). If you want to keep returning the legacy format, please set 'return_legacy_cache=True'.
Generation process: 100%|          | 1/1 [00:00<00:00, 1.76it/s]
result: ('Taylor Alison Swift is a multi-platinum, award-winning singer-songwriter and pop culture icon. Born on December 13, 1989, in Reading, 'Jack Ma is a Chinese business magnate and the founder of Alibaba Group, a multinational e-commerce company. He is one of the richest people in the world and')
⊙ (flashrag) root@autodl-container-48b1458428-957c1534:~/autodl-tmp/FlashRAG/examples/quick_start#
```

章节三：FlashRAG 模块剖析

1. RAG组件与Pipeline

在 FlashRAG 中，我们构建了一系列常见的 RAG 组件，包括检索器、生成器、优化器等。基于这些组件，我们组装了多个pipeline来实现 RAG workflow，同时也提供了灵活性，允许用户自定义组合这些组件以创建自己的pipeline。

类型	模块	描述
判断器	SKR Judger	判断是否使用 SKR 方法进行检索
检索器	Dense Retriever	使用双编码器模型（如 dpr、bge、e5），基于 faiss 进行检索
	BM25 Retriever	基于 Lucene 的稀疏检索方法
	Bi-Encoder Reranker	使用双编码器计算匹配分数
	Cross-Encoder Reranker	使用交叉编码器计算匹配分数
优化器	Extractive Refiner	通过提取重要上下文对输入进行优化
	Abstractive Refiner	通过 seq2seq 模型优化输入
	LLMLingua Refiner	LLMLingua 系列提示压缩器
	SelectiveContext Refiner	Selective-Context 提示压缩器
	KG Refiner	使用 Trace 方法构建知识图谱
生成器	Encoder-Decoder Generator	编码器-解码器模型，支持 Fusion-in-Decoder (FiD)
	Decoder-only Generator	原生的 Transformers 实现
	FastChat Generator	使用 FastChat 加速生成
	vllm Generator	使用 vllm 加速生成

1.1 检索器

检索器接收一系列查询，并从语料库中检索与每个查询对应的前 k 个文档。可以使用 `flashrag.utils.get_retriever` 加载检索器，该方法根据配置中指定的 `retrieval_method` 确定检索器的类型（BM25 或 Dense），并在内部初始化检索器所需的语料库和模型。

```
retriever = get_retriever(config)
```

然后，可以使用 `retriever.search`（针对单个查询）或 `retriever.batch_search`（针对查询列表）执行检索。

```
# 如果设置 `return_scores=True`
retrieval_results, scores = self.retriever.batch_search(input_query_list,
return_scores=True)

# 如果设置 `return_scores=False`
retrieval_results = self.retriever.batch_search(input_query_list,
return_scores=False)
```

使用 `batch_search` 时，`retrieval_results` 是一个两层嵌套列表，如下所示：


```
[
    [doc1, doc2, ...], # query1 的文档项
    [doc1, doc2, ....], # query2 的文档项
    ...
]
```

使用 search 时, retrieval_results 是一个常规列表, 包含每个查询的前 k 个文档项。每个文档项是一个包含 doc_id、title、contents 等字段的字典 (类似于语料库中的内容)。scores 的格式与 retrieval_results 相同, 只是每个 doc_item 被替换为一个 float 值, 表示检索器提供的文档与查询之间的匹配得分。

- **检索器的附加功能**: 检索器的 search 和 batch_search 函数通过两个装饰器实现了三个附加功能:
- **预加载检索结果**: 适用于你自己为查询提供一些检索结果的情况。如果配置中设置了 use_retrieval_cache 且提供了缓存文件, 它会首先检查缓存文件中是否包含相应查询的检索结果, 如果存在则直接返回。
- **保存检索结果**: 如果 save_retrieval_cache 设置为 True, 检索器会将每个查询的检索结果保存为缓存, 方便下次直接使用缓存。
- **重排序**: 如果 use_reranker=True, search 函数将集成重排序功能, 对检索结果进行进一步排序。

1.2 生成器

生成器接收提示作为输入, 并返回每个提示对应的输出。可以使用 flashrag.utils.get_generator 加载生成器。根据输入的生成器模型名称, 它会选择加载不同结构的生成器模型。

```
generator = get_generator(config)
```

然后, 使用 generate 方法进行生成。

```
input_list = ['who is taylor swift?', 'who is jack ma?']
result = generator.generate(input_list)
```

可以通过 return_scores 获取每个 token 的生成概率。

```
result, scores = generator.generate(input_list, return_scores=True)
```

generate 函数还可以接收生成所需的参数, 例如 topk、do_sample 等。这些参数也可以在配置中指定, 但在 generate 中指定的参数优先级更高。

```
result = generator.generate(input_list, top_p=1.0, max_tokens=32)
```

1.3 配置

Config 类支持使用 .yaml 文件作为输入或直接使用变量作为输入。变量的优先级高于文件。所有后续组件设置都依赖于 Config。

如果有需要使用的变量未在上述两个地方指定，将加载默认值（basic_config.yaml）。

```
from flashrag.config import Config

config_dict = {"retrieval_method": "bge"}
config = Config('my_config.yaml', config_dict = config_dict)
print(config['retrieval_method'])
```

1.4 Pipeline

我们根据推理路径将 RAG 方法分为四种类型：

- **顺序型**：按顺序执行 RAG 过程，例如 Query - (预检索) - Retriever - (后检索) - Generator。
- **条件型**：根据不同类型的输入查询执行不同的路径。
- **分支型**：并行执行多个路径，并合并每条路径的响应。
- **循环型**：迭代执行检索和生成。

在每种类别中，我们实现了相应的通用pipeline。

- **流水线组件**

类型	模块	描述
顺序型	Sequential Pipeline	按线性方式执行查询，支持优化器和重排序器
条件型	Conditional Pipeline	使用判断器模块，为不同类型的查询提供独特的执行路径
分支型	REPLUG Pipeline	通过整合多个生成路径中的概率生成答案
	SuRe Pipeline	根据每个文档的生成结果进行排序和合并
	Iterative Pipeline	交替执行检索和生成
	Self-Ask Pipeline	使用 Self-Ask 方法将复杂问题分解为子问题
循环型	Self-RAG Pipeline	自适应检索、优化和生成
	FLARE Pipeline	在生成过程中动态检索
	IRCOT Pipeline	将检索过程与 CoT (Chain-of-Thought) 集成

1. REPLUG pipeline

REPLUG: Retrieval-Augmented Black-Box Language Models

<https://arxiv.org/abs/2301.12652v4>

2. SuRe pipeline

SuRe: Summarizing Retrievals using Answer Candidates for Open-domain QA of LLMs

<https://arxiv.org/abs/2404.13081>

3. Iterative pipeline

- Enhancing Retrieval-Augmented Large Language Models with Iterative Retrieval-Generation Synergy

<https://arxiv.org/abs/2305.15294v2>

- Retrieval-Generation Synergy Augmented Large Language Models

<https://arxiv.org/abs/2310.05149>

4. Self-Ask pipeline

Measuring and Narrowing the Compositionality Gap in Language Models

<https://arxiv.org/abs/2210.03350v3>

5. Self-RAG pipeline

Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection

<https://arxiv.org/pdf/2310.11511>

6. FLARE pipeline

Active Retrieval Augmented Generation

<https://arxiv.org/abs/2305.06983v2>

7. IRCOT pipeline

Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions
<https://arxiv.org/abs/2212.10509v2>

2. 数据集和文档语料库

• 数据集

我们收集并处理了 38 个广泛用于 RAG 研究的数据集，对其进行预处理以确保格式一致，便于使用。对于某些数据集（如 Wiki-asp），我们根据社区常用的方法对其进行调整，以适配 RAG 任务的要求。

所有数据集均可在 [Huggingface](https://huggingface.co/datasets/RUC-NLPIR/FlashRAG_datasets) 获取
https://huggingface.co/datasets/RUC-NLPIR/FlashRAG_datasets

对于每个数据集，我们将每个分割保存为一个 .jsonl 文件，每一行表示一个字典，格式如下：

```
{
  'id': str,
  'question': str,
  'golden_answers': List[str],
  'metadata': dict
}
```

• 文档语料库

我们的工具支持检索文档集合的 .jsonl 格式，其结构如下：

```
{"id": "0", "contents": "..."}
{"id": "1", "contents": "..."}

```

contents 键是构建索引的关键。对于同时包含文本和标题的文档，我们建议将 contents 的值设置为 {title}\n{text}。语料库文件还可以包含其他键，用于记录文档的附加特征。

在学术研究中，**Wikipedia** 和 **MS MARCO** 是最常用的检索文档集合。对于 Wikipedia，我们提供了一个综合脚本（参考：process-wiki.md）将任意 Wikipedia 转储文件处理为干净的语料库。对于 MS MARCO，它在发布时已被处理完毕，可以直接从其 [托管链接](#) 下载到 Hugging Face 平台。

[Tevatron/msmarco-passage-corpus](https://huggingface.co/datasets/Tevatron/msmarco-passage-corpus)
<https://huggingface.co/datasets/Tevatron/msmarco-passage-corpus>

3. 参数配置

FlashRAG 提供了全面的参数管理，用于控制实验。FlashRAG 支持两种参数配置方式：**YAML 配置文件**和**参数字典**。参数通过 Config 模块进行分配。

在实际使用中，config 可以以字典访问的方式使用，例如：

```
from flashrag.config import Config
config_dict = {'generator_model': 'llama2-7B'}
config = Config(config_dict=config_dict)
model_name = config['generator_model']
```

在大多数情况下，大部分参数无需修改。

- 配置文件

配置文件应使用 **YAML** 格式，用户需根据 YAML 语法填写相应的参数。本库提供了模板文件供用户参考，模板中包含每个参数的具体含义的注释。

```
from flashrag.config import Config
config = Config(config_file_path='myconfig.yaml')
```

- 参数字典

另一种方式是通过 Python 字典设置配置，其中键为参数名称，值为参数值。相比使用文件，这种方式更灵活。

以下是使用参数字典的代码：

```
from flashrag.config import Config
config_dict = {'generator_model': 'llama2-7B'}
config = Config(config_dict=config_dict)
```

- 优先级

在 **FlashRAG** 中，支持同时使用两种方法。配置方式的优先级为：**参数字典 > 配置文件 > 默认设置**。

默认设置记录在 [basic_config.yaml](#) 中。

- 路径设置

为了方便管理各种模型路径和索引路径，我们引入了 **model2path**、**model2pooling** 和 **method2index**。用户只需在这里填写相应的名称和路径，即可自动加载需要使用的路径。

例如，在 **model2path** 中设置了 **llama2-7B** 的路径后，可以直接将其指定为对应的模型，而无需重写路径。

```
model2path:
  llama2-7B: 'my_path'
generator_model: 'llama2-7B'
```

如果在字典中找不到相应的模型路径，它会查找与模型路径对应的参数（例如，generator_model_path）。

此规则适用于以下参数：retrieval_method, rerank_model_name, generator_model, index_path

• 配置示例

示例配置文件可以在 basic_config.yaml 中找到。下面我们将介绍配置文件中的每个部分。

basic_config.yaml :

```
# 基础设置

# -----全局路径-----
-----#

# 各种模型的路径
model2path:
  e5: "intfloat/e5-base-v2"
  bge: "intfloat/e5-base-v2"
  contriever: "facebook/contriever"
  llama2-7B-chat: "meta-llama/Llama-2-7b-chat-hf"
  llama2-7B: "meta-llama/Llama-2-7b-hf"
  llama2-13B: "meta-llama/Llama-2-13b-hf"
  llama2-13B-chat: "meta-llama/Llama-2-13b-chat-hf"

# 每个嵌入模型的池化方法
model2pooling:
  e5: "mean"
  bge: "cls"
  contriever: "mean"
  jina: "mean"
  dpr: cls

# 检索模型的索引路径
method2index:
  e5: ~ # 如果未提供，将自动设置
  bm25: ~
  contriever: ~

# -----环境设置-----
-----#

# 数据和输出的目录路径
data_dir: "dataset/"
save_dir: "output/"

gpu_id: "0,1,2,3"
dataset_name: "nq" # 数据集名称，存储在 data_dir 中
```

```

split: ["test"] # 要加载的数据集分割 (例如 train, dev, test)

# 测试的采样配置
test_sample_num: ~ # 测试的样本数量 (仅适用于 dev/test 分割), 如果为 None, 则测试所有样本
random_sample: False # 是否随机抽取测试样本

# 用于结果复现的随机种子
seed: 2024

# 是否保存中间数据
save_intermediate_data: True
save_note: 'experiment'

# -----检索设置-----
-----#
# 如果设置了名称, 模型路径将在全局路径中查找
retrieval_method: "e5" # 检索模型的名称或路径
retrieval_model_path: ~ # 检索模型的路径
index_path: ~ # 如果未提供, 将自动设置
faiss_gpu: False # 是否使用 GPU 保存索引
corpus_path: ~ # 以 '.jsonl' 格式存储文档的语料库路径

use_sentence_transformer: False # 如果设置, 检索器将通过 `sentence transformer` 库加载
retrieval_topk: 5 # 检索的文档数量
retrieval_batch_size: 256 # 检索的批处理大小
retrieval_use_fp16: True # 是否对检索模型使用 fp16
retrieval_query_max_length: 128 # 查询的最大长度
save_retrieval_cache: True # 是否保存检索缓存
use_retrieval_cache: False # 是否使用检索缓存
retrieval_cache_path: ~ # 检索缓存的路径
retrieval_pooling_method: ~ # 如果未提供, 将自动设置

use_reranker: False # 是否使用重排序器
rerank_model_name: ~ # 与 retrieval_method 相同
rerank_model_path: ~ # 重排序器模型的路径, 路径将在 `model2path` 中自动查找
rerank_pooling_method: ~
rerank_topk: 5 # 重排序后保留的文档数量
rerank_max_length: 512
rerank_batch_size: 256 # 重排序器的批处理大小
rerank_use_fp16: True

# -----生成器设置-----
-----#
framework: fschat # 大语言模型的推理框架, 支持: 'hf', 'vllm', 'fschat', 'openai'
generator_model: "llama3-8B-instruct" # 生成器模型的名称或路径
# openai 模型的设置, 仅在 openai 框架中有效
openai_setting:
    api_key: ~
    base_url: ~

generator_model_path: ~

```



```
generator_max_input_len: 1024 # 输入的最大长度
generator_batch_size: 4 # 生成的批处理大小, 对 vllm 无效
generation_params:
  #do_sample: false
  max_tokens: 32
  #temperature: 1.0
  #top_p: 1.0
use_fid: False # 是否使用 FID, 仅对编码器-解码器模型有效

# -----评估设置-----
# -----#
# 评估结果的指标
metrics: ['em', 'f1', 'acc', 'precision', 'recall']
# 指定指标的设置, 将在某些指标中调用
metric_setting:
  retrieval_recall_topk: 5
save_metric_score: True # 是否将指标得分保存到 txt 文件中
```

章节四：构建索引实践

1. 准备语料库

为了构建索引，首先需要将语料库保存为如下所示的 jsonl 格式，每行代表一个文档。

```
{"id": "0", "contents": "用于构建索引的内容"}
{"id": "1", "contents": "用于构建索引的内容"}
```

如果你希望使用维基百科作为语料库，可以参考文档【FlashRAG/docs/process-wiki.md】将其转换为索引格式。

注意 ⚠️ 可以基于OpenAI API进行数据抽取，理解，生成。

2. 创建索引

- 对于 稠密检索方法（尤其是流行的嵌入模型），使用 faiss 构建索引。
- 对于 稀疏检索方法（BM25），基于 Pyserini 或 bm25s 将语料库构建为 Lucene 倒排索引。构建的索引包含原始文档。

(1) 对于稠密检索方法

(1.1) 基于Transformers库

```
python -m flashrag.retriever.index_builder \
    --retrieval_method e5 \
    --model_path /root/autodl-tmp/models/e5-base-v2 \
    --corpus_path /root/autodl-tmp/FlashRAG/examples/quick_start/indexes/docs.jsonl \
    --save_dir /root/autodl-tmp/FlashRAG/examples/quick_start/indexes/ \
    --use_fp16 \
    --max_length 512 \
    --batch_size 256 \
    --pooling_method mean \
    --faiss_type Flat
```

```
● (flashrag) (base) root@autodl-container-48b1458428-957c1534:~/autodl-tmp# python -m flashrag.retriever.index_builder \
--retrieval_method e5 \
--model_path /root/autodl-tmp/models/e5-base-v2 \
--corpus_path /root/autodl-tmp/FlashRAG/examples/quick_start/indexes/docs.jsonl \
--save_dir /root/autodl-tmp/FlashRAG/examples/quick_start/indexes/ \
--use_fp16 \
--max_length 512 \
--batch_size 256 \
--pooling_method mean \
--faiss_type Flat
/root/autodl-tmp/FlashRAG/flashrag/retriever/index_builder.py:63: UserWarning: Instruction is set to default: passage:
warnings.warn(f"Instruction is set to default: {self.instruction}")
/root/autodl-tmp/FlashRAG/examples/quick_start/indexes/
/root/autodl-tmp/FlashRAG/flashrag/retriever/index_builder.py:103: UserWarning: Some files already exists in save dir and may be overwri
ten.
warnings.warn("Some files already exists in save dir and may be overwritten.", UserWarning)
Generating train split: 20 examples [00:00, 1827.34 examples/s]
Finish loading...
Inference Embeddings:: 100% | 1/1 [00:00<00:00, 2.42it/s]
Creating index
Finish!
```

--instruction: 某些嵌入模型需要额外的指令以在编码前连接查询，可以在此处指定。目前，对于 E5 和 BGE 模型，我们将自动填充指令，而其他模型需要手动补充。

--faiss_type: Flat表示使用扁平索引（全精度暴力搜索），Flat 索引虽然精确，但可能在大规模数据集上速度较慢。其他常用索引类型包括 IVF、HNSW 等。

```
python -m flashrag.retriever.index_builder \
    --retrieval_method bge \
    --model_path /root/autodl-tmp/models/bge-large-zh-v1.5 \
    --corpus_path /root/autodl-
tmp/FlashRAG/examples/quick_start/indexes/docs.jsonl \
    --save_dir /root/autodl-tmp/FlashRAG/examples/quick_start/indexes/ \
    --use_fp16 \
    --max_length 512 \
    --batch_size 256 \
    --pooling_method mean \
    --faiss_type Flat
```

- 执行命令：

```
python -m flashrag.retriever.index_builder \
    --retrieval_method e5 \
    --model_path /root/autodl-tmp/models/e5-base-v2 \
    --corpus_path /root/autodl-tmp/FlashRAG/examples/quick_start/indexes/docs.jsonl \
    --save_dir /root/autodl-tmp/FlashRAG/examples/quick_start/indexes/ \
    --use_fp16 \
    --max_length 512 \
    --batch_size 256 \
    --sentence_transformer \
    --faiss_type Flat
```

```
python -m flashrag.retriever.index_builder \
  --retrieval_method bm25 \
  --corpus_path /root/autodl-
tmp/FlashRAG/examples/quick_start/indexes/docs.jsonl \
  --bm25_backend bm25s \
  --save_dir /root/autodl-tmp/FlashRAG/examples/quick_start/indexes/
```

© 2026 TGLTommy 原创出品 · All Rights Reserved

• 使用 Pyserini 构建索引

Ubuntu系统, 需要安装 JAVA

sudo apt install openjdk-21-jdk

```
python -m flashrag.retriever.index_builder \
    --retrieval_method bm25 \
    --corpus_path /root/autodl-
tmp/FlashRAG/examples/quick_start/indexes/docs.jsonl \
    --bm25_backend pyserini \
    --save_dir /root/autodl-tmp/FlashRAG/examples/quick_start/indexes/
```

```
2024-12-16 22:31:32,140 INFO [main] index.AbstractIndexer (AbstractIndexer.java:205) - Setting log level to INFO
2024-12-16 22:31:32,143 INFO [main] index.AbstractIndexer (AbstractIndexer.java:208) - ===== Loading Index Configuration =====
2024-12-16 22:31:32,143 INFO [main] index.AbstractIndexer (AbstractIndexer.java:209) - AbstractIndexer settings:
2024-12-16 22:31:32,143 INFO [main] index.AbstractIndexer (AbstractIndexer.java:210) - + DocumentCollection path: /root/autodl-tmp/FlashRAG/examples/quick_start/indexes/bm25/temp
2024-12-16 22:31:32,143 INFO [main] index.AbstractIndexer (AbstractIndexer.java:211) - + CollectionClass: JsonCollection
2024-12-16 22:31:32,143 INFO [main] index.AbstractIndexer (AbstractIndexer.java:212) - + Index path: /root/autodl-tmp/FlashRAG/examples/quick_start/indexes/bm25
2024-12-16 22:31:32,144 INFO [main] index.AbstractIndexer (AbstractIndexer.java:213) - + Threads: 1
2024-12-16 22:31:32,144 INFO [main] index.AbstractIndexer (AbstractIndexer.java:214) - + Optimize (merge segments)? false
Dec 16, 2024 10:31:32 PM org.apache.lucene.store.MemorySegmentIndexInputProvider <init>
INFO: Using MemorySegmentIndexInput with Java 21; to disable start with -Dorg.apache.lucene.store.MMapDirectory.enableMemorySegments=false
2024-12-16 22:31:32,166 INFO [main] index.IndexCollection (IndexCollection.java:246) - Using DefaultEnglishAnalyzer
2024-12-16 22:31:32,166 INFO [main] index.IndexCollection (IndexCollection.java:247) - Stemmer: porter
2024-12-16 22:31:32,166 INFO [main] index.IndexCollection (IndexCollection.java:248) - Keep stopwords? false
2024-12-16 22:31:32,166 INFO [main] index.IndexCollection (IndexCollection.java:249) - Stopwords file: null
2024-12-16 22:31:32,254 INFO [main] index.IndexCollection (IndexCollection.java:197) - IndexCollection settings:
2024-12-16 22:31:32,254 INFO [main] index.IndexCollection (IndexCollection.java:198) - + Generator: DefaultLuceneDocumentGenerator
2024-12-16 22:31:32,254 INFO [main] index.IndexCollection (IndexCollection.java:199) - + Language: en
2024-12-16 22:31:32,255 INFO [main] index.IndexCollection (IndexCollection.java:200) - + Stemmer: porter
2024-12-16 22:31:32,255 INFO [main] index.IndexCollection (IndexCollection.java:201) - + Keep stopwords? false
2024-12-16 22:31:32,255 INFO [main] index.IndexCollection (IndexCollection.java:202) - + Stopwords: null
2024-12-16 22:31:32,255 INFO [main] index.IndexCollection (IndexCollection.java:203) - + Store positions? false
2024-12-16 22:31:32,255 INFO [main] index.IndexCollection (IndexCollection.java:204) - + Store docvectors? false
2024-12-16 22:31:32,256 INFO [main] index.IndexCollection (IndexCollection.java:205) - + Store document "contents" field? false
2024-12-16 22:31:32,256 INFO [main] index.IndexCollection (IndexCollection.java:206) - + Store document "raw" field? false
2024-12-16 22:31:32,256 INFO [main] index.IndexCollection (IndexCollection.java:207) - + Additional fields to index: []
2024-12-16 22:31:32,256 INFO [main] index.IndexCollection (IndexCollection.java:208) - + Whitelist: null
2024-12-16 22:31:32,256 INFO [main] index.IndexCollection (IndexCollection.java:209) - + Pretokenized?: false
2024-12-16 22:31:32,256 INFO [main] index.IndexCollection (IndexCollection.java:210) - + Codec: Lucene99
2024-12-16 22:31:32,256 INFO [main] index.AbstractIndexer (AbstractIndexer.java:238) - ===== Indexing Collection =====
2024-12-16 22:31:32,258 INFO [main] index.AbstractIndexer (AbstractIndexer.java:247) - Thread pool with 1 threads initialized.
2024-12-16 22:31:32,259 INFO [main] index.AbstractIndexer (AbstractIndexer.java:248) - 1 file found in /root/autodl-tmp/FlashRAG/examples/quick_start/indexes/bm25/temp
2024-12-16 22:31:32,259 INFO [main] index.AbstractIndexer (AbstractIndexer.java:249) - Starting to index...
2024-12-16 22:31:32,412 INFO [main] index.AbstractIndexer (AbstractIndexer.java:307) - Indexing Complete! 20 documents indexed
2024-12-16 22:31:32,412 INFO [main] index.AbstractIndexer (AbstractIndexer.java:308) - ===== Final Counter Values =====
2024-12-16 22:31:32,412 INFO [main] index.AbstractIndexer (AbstractIndexer.java:309) - indexed: 20
2024-12-16 22:31:32,413 INFO [main] index.AbstractIndexer (AbstractIndexer.java:310) - -unindexable: 0
2024-12-16 22:31:32,413 INFO [main] index.AbstractIndexer (AbstractIndexer.java:311) - -empty: 0
2024-12-16 22:31:32,413 INFO [main] index.AbstractIndexer (AbstractIndexer.java:312) - -skipped: 0
2024-12-16 22:31:32,413 INFO [main] index.AbstractIndexer (AbstractIndexer.java:313) - -errors: 0
2024-12-16 22:31:32,417 INFO [main] index.AbstractIndexer (AbstractIndexer.java:316) - Total 20 documents indexed in 00:00:00
Finish!
```

3. 案例Demo

```
streamlit run demo_zh.py --server.port 6006 --server.address 0.0.0.0
```

FlashRAG Demo

This demo retrieves documents and generates responses based on user input.

Enter your prompt:

给我介绍一下监督学习的概念

Generate Responses

References

[1]: 监督学习

监督学习是一种机器学习方法，通过带标签的数据对模型进行训练。它包括分类和回归两种主要任务。模型通过不断调整参数以最小化预测误差，常见算法包括线性回归、支持向量机和神经网络。

[2]: 无监督学习

无监督学习处理无标签数据，尝试发现数据中的模式或分组。常用方法包括聚类算法（如K均值、层次聚类）和降维算法（如主成分分析PCA）。它在数据探索和特征工程中发挥重要作用。

章节五：论文实验流程复现

所有使用的代码都基于仓库的 [example/methods](#)。我们为各种方法设置了适当的超参数。如果你需要自己调整它们，可以参考为每种方法提供的配置字典以及每种方法的原始论文。

1. 设置基本配置

首先，你需要在 `my_config.yaml` 中填写各种下载的路径。具体来说，你需要填写以下四个字段：

- **model2path**: 用你自己的路径替换 E5 和 Llama3-8B-instruct 模型的路径
- **method2index**: 填写使用 E5 构建的索引文件的路径
- **corpus_path**: 填写 `jsonl` 格式的 Wikipedia 语料库文件的路径
- **data_dir**: 更改为你自己的数据集下载路径


```

(tflashrag) (base) root@autodl-container-48b1458428-957c1534:~/autodl-tmp/tflashrag/examples/methods#
(flashrag) (base) root@autodl-container-48b1458428-957c1534:~/autodl-tmp/FlashRAG/examples/methods# python -m flashrag.retriever.index_builder \
--retrieval_method aar-ance \
--model_path /root/autodl-tmp/models/AAR-ANCE \
--corpus_path /root/autodl-tmp/FlashRAG/examples/quick_start/indexes/general_knowledge.jsonl \
--save_dir /root/autodl-tmp/FlashRAG/examples/quick_start/indexes/ \
--use_fp16 \
--max_length 512 \
--batch_size 256 \
--pooling_method cls \
--faiss_type Flat
/root/autodl-tmp/FlashRAG/flashrag/retriever/index_builder.py:61: UserWarning: Instruction is not set!
warnings.warn("Instruction is not set!")
/root/autodl-tmp/FlashRAG/examples/quick_start/indexes/
/root/autodl-tmp/FlashRAG/flashrag/retriever/index_builder.py:103: UserWarning: Some files already exists in save dir and may be overwritten.
warnings.warn("Some files already exists in save dir and may be overwritten.", UserWarning)
Finish loading...
You are using the default legacy behaviour of the <class 'transformers.models.t5.tokenization_t5.T5Tokenizer'>. This is expected, and simply means that the 'legacy' (previous
) behavior will be used so nothing changes for you. If you want to use the new behaviour, set 'legacy=False'. This should only be set if you understand what it means, and tho
roughly read the reason why this was added as explained in https://github.com/huggingface/transformers/pull/24565
Inference Embeddings: 0% | 0/59 [00:00<7, ?it/s]
Passing a tuple of 'past_key_values' is deprecated and will be removed in Transformers v4.48.0. You should pass an instance of 'EncoderDecoderCache' instead, e.g. 'past_key_v
alues=EncoderDecoderCache.from_legacy_cache(past_key_values)'.
Inference Embeddings: 100% | 59/59 [00:45<00:00, 1.30it/s]
Creating index
Finish!

```

• 步骤3: 修改 methods/my_config.yaml 里的配置信息

```

# -----Global Paths-----#
# Paths to models
model2path:
  e5: "intfloat/e5-base-v2"
  bge: "BAAI/bge-base-en-v1.5"
  llama2-7B-chat: "meta-llama/Llama-2-7b-chat-hf"
  llama2-7B: "meta-llama/Llama-2-7b-hf"
  llama2-13B: "meta-llama/Llama-2-13b-hf"
  llama2-13B-chat: "meta-llama/Llama-2-13b-chat-hf"
  llama3-8B-instruct: "meta-llama/Meta-Llama-3-8B-Instruct"
  Llama-3.2-1B-Instruct: "/root/autodl-tmp/models/Llama-3.2-1B-Instruct"

# Pooling methods for each embedding model
model2pooling:
  e5: "mean"
  bge: "cls"
  contriever: "mean"
  jina: 'mean'
  dpr: 'cls'

# Indexes path for retrieval models
method2index:
  e5: ~
  bm25: ~
  contriever: ~

# -----Environment Settings-----#
# Directory paths for data and outputs
data_dir: "/root/autodl-tmp/datasets/FlashRAG_datasets"
save_dir: "output/"

gpu_id: "0,1,2,3"
dataset_name: "nq" # name of the dataset in data_dir
split: [ "test" ] # dataset split to load (e.g. train,dev,test)

# Sampling configurations for testing

```



```

test_sample_num: ~ # number of samples to test (only work in dev/test split),
if None, test all samples
random_sample: False # whether to randomly sample the test samples

# Seed for reproducibility
seed: 2024

# Whether save intermediate data
save_intermediate_data: True
save_note: 'experiment'

# -----Retrieval Settings-----
---#
# If set the name, the model path will be find in global paths
retrieval_method: "contriever" # name or path of the retrieval model.
index_path: ~ # SET AUTOMATICALLY IF NOT PROVIDED.
faiss_gpu: True # whether use gpu to hold index
corpus_path: "/root/autodl-
tmp/FlashRAG/examples/quick_start/indexes/general_knowledge.jsonl"

instruction: ~ # instruction for retrieval model
retrieval_topk: 5 # number of retrieved documents
retrieval_batch_size: 256 # batch size for retrieval
retrieval_use_fp16: True # whether to use fp16 for retrieval model
retrieval_query_max_length: 128 # max length of the query
save_retrieval_cache: False # whether to save the retrieval cache
use_retrieval_cache: False # whether to use the retrieval cache
retrieval_cache_path: ~ # path to the retrieval cache
retrieval_pooling_method: ~ # set automatically if not provided

use_reranker: False # whether to use reranker
rerank_model_name: ~ # same as retrieval_method
rerank_model_path: ~ # path to reranker model, path will be automatically find
in `retriever_model2path`
rerank_pooling_method: ~
rerank_topk: 5 # number of remain documents after reranking
rerank_max_length: 512
rerank_batch_size: 256 # batch size for reranker
rerank_use_fp16: True

# -----Generator Settings-----#
framework: vllm # inference frame work of LLM, supporting:
'hf', 'vllm', 'fschat'
generator_model: "Llama-3.2-1B-Instruct" # name or path of the generator model
generator_max_input_len: 2048 # max length of the input
generator_batch_size: 2 # batch size for generation, invalid for vllm
generation_params:
  do_sample: False
  max_tokens: 32
use_fid: False # whether to use FID, only valid in encoder-decoder model

```

```

# -----Refiner Settings-----
--#
# If set, the refiner will be used to refine the retrieval documents.
refiner_name: ~
refiner_model_path: ~

# Used for extractive method (e.g embedding models)
refiner_topk: 5 # number of remain sentence after refiner
refiner_pooling_method: 'mean' # pooling method of refiner model
refiner_encode_max_length: 256
# Used for abstractive method (e.g. generation models like bart-large-cnn)
refiner_max_input_length: 1024
refiner_max_output_length: 512

# Specify settings for llmlingua
llmlingua_config:
  rate: 0.55
  condition_in_question: 'after_condition'
  reorder_context: 'sort'
  dynamic_context_compression_ratio: 0.3
  condition_compare: True
  context_budget: "+100"
  rank_method: 'longllmlingua'
sc_config:
  'reduce_ratio': 0.5

# -----Evaluation Settings-----#
# Metrics to evaluate the result
metrics: [ 'em', 'f1', 'acc', 'precision', 'recall' ]
# Specify setting for metric, will be called within certain metrics
metric_setting:
  retrieval_recall_topk: 5
save_metric_score: True # whether to save the metric score into txt file

```

- 步骤4: 在 `run_exp.py` 中的 `AAR` 函数中修改 `index_path` 和 `model2path`

```

def aar(args):
    """
    Reference:
        Zichun Yu et al. "Augmentation-Adapted Retriever Improves
        Generalization of Language Models as Generic Plug-In"
        in ACL 2023.
        Official repo: https://github.com/OpenMatch/Augmentation-Adapted-
        Retriever
    """
    # two types of checkpoint: ance / contriever
    # retrieval_method = "AAR-contriever" # AAR-ANCE
    # index path of this retriever
    retrieval_method = args.method_name

```

```

if "contriever" in retrieval_method:
    index_path = "/root/autodl-
tmp/FlashRAG/examples/quick_start/indexes/aar-contriever_Flat.index"
else:
    index_path = "aar-ance_Flat.index"

model2path = {"AAR-contriever": "/root/autodl-tmp/models/AAR-Contriever-
KILT", "AAR-ANCE": "model/AAR-ANCE"}
model2pooling = {"AAR-contriever": "mean", "AAR-ANCE": "cls"}
save_note = retrieval_method
config_dict = {
    "retrieval_method": retrieval_method,
    "model2path": model2path,
    "index_path": index_path,
    "model2pooling": model2pooling,
    "save_note": save_note,
    "gpu_id": args.gpu_id,
    "dataset_name": args.dataset_name,
}

# preparation
config = Config("/root/autodl-
tmp/FlashRAG/examples/methods/my_config.yaml", config_dict)
all_split = get_dataset(config)
test_data = all_split[args.split]

from flashrag.pipeline import SequentialPipeline

pred_process_fun = lambda x: x.split("\n")[0]
pipeline = SequentialPipeline(config)
# result = pipeline.run(test_data, pred_process_fun=pred_process_fun)
result = pipeline.run(test_data)
print(result)

```

• 步骤5: 执行命令

```

# AAR-contriever

python run_exp.py \
    --method_name 'AAR-contriever' \
    --split 'test' \
    --dataset_name 'nq' \
    --gpu_id '0'

```

```

Retrieval process: 0%| 0/15 [00:00<7, 7it/s]
Use `` as retrieval instruction
Retrieval process: 7%| 1/15 [00:00<00:05, 2.70it/s]
Use `` as retrieval instruction
Retrieval process: 13%| 2/15 [00:00<00:04, 2.69it/s]
Use `` as retrieval instruction
Retrieval process: 27%| 4/15 [00:00<00:02, 5.08it/s]
Use `` as retrieval instruction
Retrieval process: 40%| 6/15 [00:01<00:01, 6.76it/s]
Use `` as retrieval instruction
Retrieval process: 53%| 8/15 [00:01<00:00, 7.97it/s]
Use `` as retrieval instruction
Retrieval process: 67%| 10/15 [00:01<00:00, 8.78it/s]
Use `` as retrieval instruction
Retrieval process: 80%| 12/15 [00:01<00:00, 9.40it/s]
Use `` as retrieval instruction
Retrieval process: 93%| 14/15 [00:01<00:00, 9.82it/s]
Use `` as retrieval instruction
Retrieval process: 100%| 15/15 [00:01<00:00, 8.01it/s]
Processed prompts: 100%| 3610/3610 [01:00<00:00, 50.80it/s, est. speed input: 21694.83 toks/s, output: 528.47 toks/s]
{'em': 0.0362808864265928, 'f1': 0.07238351310919992, 'acc': 0.08282548476454293, 'precision': 0.06880447959877058, 'recall': 0.11102954755309317}
Dataset 'nq' with 3610 items
[rank0]:[W121/ 12:50:11.915/76945 ProcessGroupNCCL.cpp:1250] Warning: WARNING: process group has NOT been destroyed before we destruct ProcessGroupNCCL. On normal program exit, the application should call destroy_process_group to ensure that any pending NCCL operations have finished in this process. In rare cases this process can exit before this point and block the progress of another member of the process group. This constraint has always been present, but this warning has only been added since PyTorch 2.4 (function operator())

```

AAR-ANCE

```

python run_exp.py \
--method_name 'AAR-ANCE' \
--split 'test' \
--dataset_name 'nq' \
--gpu_id '0'

```

```

Retrieval process: 13%| 2/15 [00:00<00:03, 4.11it/s]
Use `` as retrieval instruction
Retrieval process: 20%| 3/15 [00:00<00:02, 5.43it/s]
Use `` as retrieval instruction
Retrieval process: 27%| 4/15 [00:00<00:01, 6.29it/s]
Use `` as retrieval instruction
Retrieval process: 33%| 5/15 [00:00<00:01, 6.91it/s]
Use `` as retrieval instruction
Retrieval process: 40%| 6/15 [00:01<00:01, 7.29it/s]
Use `` as retrieval instruction
Retrieval process: 47%| 7/15 [00:01<00:01, 7.55it/s]
Use `` as retrieval instruction
Retrieval process: 53%| 8/15 [00:01<00:00, 7.77it/s]
Use `` as retrieval instruction
Retrieval process: 60%| 9/15 [00:01<00:00, 8.02it/s]
Use `` as retrieval instruction
Retrieval process: 67%| 10/15 [00:01<00:00, 8.21it/s]
Use `` as retrieval instruction
Retrieval process: 73%| 11/15 [00:01<00:00, 8.55it/s]
Use `` as retrieval instruction
Retrieval process: 80%| 12/15 [00:01<00:00, 8.77it/s]
Use `` as retrieval instruction
Retrieval process: 87%| 13/15 [00:01<00:00, 8.96it/s]
Use `` as retrieval instruction
Retrieval process: 93%| 14/15 [00:01<00:00, 9.08it/s]
Use `` as retrieval instruction
Retrieval process: 100%| 15/15 [00:01<00:00, 7.68it/s]
Processed prompts: 100%| 3610/3610 [00:56<00:00, 63.62it/s, est. speed input: 22001.73 toks/s, output: 572.01 toks/s]
{'em': 0.03102493074792244, 'f1': 0.06845410623079663, 'acc': 0.08337950138504155, 'precision': 0.06402379139384742, 'recall': 0.11054939981532776}
Dataset 'nq' with 3610 items
[rank0]:[W121/ 13:40:18.07610/506 ProcessGroupNCCL.cpp:1250] Warning: WARNING: process group has NOT been destroyed before we destruct ProcessGroupNCCL. On normal program exit, the application should call destroy_process_group to ensure that any pending NCCL operations have finished in this process. In rare cases this process can exit before this point and block the progress of another member of the process group. This constraint has always been present, but this warning has only been added since PyTorch 2.4 (function operator())

```

3. 实践-2：流水线 Self-RAG

- 第1步：修改在 `run_exp.py` 中的 `selfrag` 函数中修改 `index_path` 和 `model2path`

```
def selfrag(args):
```

```
    """
```

```
    Reference:
```

```
        Akari Asai et al. " SELF-RAG: Learning to Retrieve, Generate and Critique through self-reflection"
```

```
        in ICLR 2024.
```

```
        Official repo: https://github.com/AkariAsai/self-rag
```

```
    """
```

```
    config_dict = {
```

```

    "index_path": "/root/autodl-
tmp/FlashRAG/examples/quick_start/indexes/bge_Flat.index",
    "retrieval_model_path": "/root/autodl-tmp/models/bge-large-zh-v1.5",
    "generator_model": "selfrag-llama2-7B",
    "generator_model_path": "/root/autodl-tmp/models/selfrag_llama2_7b",
    "framework": "vllm",
    "save_note": "self-rag",
    "gpu_id": args.gpu_id,
    "generation_params": {
        "max_tokens": 128,
        "temperature": 0.0,
        "top_p": 1.0,
        "skip_special_tokens": False,
    },
    "dataset_name": args.dataset_name,
}
config = Config("/root/autodl-
tmp/FlashRAG/examples/methods/my_config.yaml", config_dict)

all_split = get_dataset(config)
test_data = all_split[args.split]

from flashrag.pipeline import SelfRAGPipeline

pipeline = SelfRAGPipeline(
    config,
    threhsold=0.2,
    max_depth=2,
    beam_width=2,
    w_rel=1.0,
    w_sup=1.0,
    w_use=1.0,
    use_grounding=True,
    use_utility=True,
    use_seqscores=True,
    ignore_cont=True,
    mode="adaptive_retrieval",
)
result = pipeline.run(test_data, long_form=False)

print(result)

```

执行命令:

```

python run_exp.py \
    --method_name 'selfrag' \
    --split 'test' \
    --dataset_name 'nq' \
    --gpu_id '0'

```

```
WARNING 12-18 16:45:55 scheduler.py:1481] Sequence group 12854 is preempted by PreemptionMode.RECOMPUTE mode because there is not enough KV cache space. This can
affect the end-to-end performance. Increase gpu_memory_utilization or tensor_parallel_size to provide more KV cache memory. total_num_cumulative_preemption=501
Processed prompts: 56% | 9985/17946 [04:27-03:36, 36.83it/s, est. speed input: 3413.92 toks/s, output: 387.65 toks/s]
WARNING 12-18 16:46:18 scheduler.py:1481] Sequence group 13724 is preempted by PreemptionMode.RECOMPUTE mode because there is not enough KV cache space. This can
affect the end-to-end performance. Increase gpu_memory_utilization or tensor_parallel_size to provide more KV cache memory. total_num_cumulative_preemption=551
Processed prompts: 61% | 10968/17946 [04:53-02:44, 42.42it/s, est. speed input: 3417.39 toks/s, output: 387.95 toks/s]
WARNING 12-18 16:46:44 scheduler.py:1481] Sequence group 14714 is preempted by PreemptionMode.RECOMPUTE mode because there is not enough KV cache space. This can
affect the end-to-end performance. Increase gpu_memory_utilization or tensor_parallel_size to provide more KV cache memory. total_num_cumulative_preemption=601
Processed prompts: 68% | 12127/17946 [05:23-02:13, 43.47it/s, est. speed input: 3432.01 toks/s, output: 387.81 toks/s]
WARNING 12-18 16:47:14 scheduler.py:1481] Sequence group 15888 is preempted by PreemptionMode.RECOMPUTE mode because there is not enough KV cache space. This can
affect the end-to-end performance. Increase gpu_memory_utilization or tensor_parallel_size to provide more KV cache memory. total_num_cumulative_preemption=651
Processed prompts: 73% | 13046/17946 [05:46-01:57, 41.88it/s, est. speed input: 3437.11 toks/s, output: 386.22 toks/s]
WARNING 12-18 16:47:38 scheduler.py:1481] Sequence group 16794 is preempted by PreemptionMode.RECOMPUTE mode because there is not enough KV cache space. This can
affect the end-to-end performance. Increase gpu_memory_utilization or tensor_parallel_size to provide more KV cache memory. total_num_cumulative_preemption=701
Processed prompts: 78% | 14054/17946 [06:12-01:39, 39.29it/s, est. speed input: 3444.05 toks/s, output: 384.14 toks/s]
WARNING 12-18 16:48:04 scheduler.py:1481] Sequence group 17806 is preempted by PreemptionMode.RECOMPUTE mode because there is not enough KV cache space. This can
affect the end-to-end performance. Increase gpu_memory_utilization or tensor_parallel_size to provide more KV cache memory. total_num_cumulative_preemption=751
Processed prompts: 84% | 15102/17946 [06:40-01:06, 42.47it/s, est. speed input: 3442.03 toks/s, output: 383.57 toks/s]
WARNING 12-18 16:48:31 scheduler.py:1481] Sequence group 18849 is preempted by PreemptionMode.RECOMPUTE mode because there is not enough KV cache space. This can
affect the end-to-end performance. Increase gpu_memory_utilization or tensor_parallel_size to provide more KV cache memory. total_num_cumulative_preemption=801
Processed prompts: 89% | 15973/17946 [07:04-00:50, 39.29it/s, est. speed input: 3439.78 toks/s, output: 386.00 toks/s]
WARNING 12-18 16:48:55 scheduler.py:1481] Sequence group 19720 is preempted by PreemptionMode.RECOMPUTE mode because there is not enough KV cache space. This can
affect the end-to-end performance. Increase gpu_memory_utilization or tensor_parallel_size to provide more KV cache memory. total_num_cumulative_preemption=851
Processed prompts: 95% | 16993/17946 [07:30-00:24, 39.70it/s, est. speed input: 3444.12 toks/s, output: 385.32 toks/s]
WARNING 12-18 16:49:21 scheduler.py:1481] Sequence group 20753 is preempted by PreemptionMode.RECOMPUTE mode because there is not enough KV cache space. This can
affect the end-to-end performance. Increase gpu_memory_utilization or tensor_parallel_size to provide more KV cache memory. total_num_cumulative_preemption=901
Processed prompts: 100% | 17946/17946 [07:54-00:00, 37.84it/s, est. speed input: 3454.87 toks/s, output: 387.88 toks/s]
{'em': 0.14515235457063713, 'f1': 0.237072178604597, 'acc': 0.2227146814404432, 'precision': 0.23521610228840545, 'recall': 0.2851061865189295}
Dataset 'nq' with 3610 items
[rank0]:[W12-18 16:49:49.208384845 ProcessGroupNCCL.cpp:1250] Warning: WARNING: process group has NOT been destroyed before we destruct ProcessGroupNCCL. On normal
program exit, the application should call destroy_process_group to ensure that any pending NCCL operations have finished in this process. In rare cases this proc
ess can exit before this point and block the progress of another member of the process group. This constraint has always been present, but this warning has only
been added since PyTorch 2.4 (function operator())
```

章节六：核心源码解析

Debug-1：simple_pipeline.py

AI进阶学习

B站课堂：

<https://space.bilibili.com/474347248/pugv>

<https://space.bilibili.com/3546748815411393/pugv>

官方网站：<https://www.tgltommy.com/> (国内访问需科学上网)

TGLTommy

Browse productsHomeContactAbout MeSign upLog in

Categories

Premium CoursesPublic Courses

DeepSeek系统实战

架构·算法·工程·应用

唐国梁Tommy | TGLTommy

DeepSeek系统实战：架构·算法·工程·应用

这是一门面向真实工程落地的大模型系统课程，围绕 DeepSeek 的技术路线，构建从“可控部署与应用搭建”到“核心机制与对齐算法”，再到“系统...

Course · By 唐国梁Tommy

Learn more

多模态大模型

前沿算法与实战应用

第一季 图文与视频理解

唐国梁Tommy

多模态大模型 前沿算法与实战应用（第一季：图文与视频理解）

本课程是一门面向多模态大模型前沿技术的系列课程，深入探讨了当前多模态大模型的核心技术与最新进展。课程覆盖了从基础概念到高级算法实现...

Course · By 唐国梁Tommy

Learn more

大模型Llama架构

从理论到实战

核心技术与底层原理全解析

唐国梁Tommy

大模型Llama架构：从理论到实战

本课程对 Llama 架构的核心模块进行了深入解析，涵盖算法原理和技术实现。你将学习如何对 Llama模型进行自定义微调、以及本地化模型推...

Course · By 唐国梁Tommy

Learn more

大模型公开课

RAG/Agent/多模态

唐国梁Tommy

大模型公开课（持续更新）

欢迎你加入我的免费公开课~我将在这里将持续分享最新AI前沿技术内容，以简明、实用的方式带你快速了解当下热门的人工智能方法与应用。课程...

Course · By 唐国梁Tommy

Learn more

深入LLM与RAG

原理、实现与应用

大语言模型LLM与检索增强生成RAG理论与实战

唐国梁Tommy

深入LLM与RAG 原理、实现与应用

本课程全面聚焦于检索增强生成 (RAG) 的理论与实践，深入讲解如何结合主流的大语言模型 (LLMs) 来实现智能信息检索和生成。课程从项目环境配...

Course · By 唐国梁Tommy

Learn more

如果你希望系统掌握大模型核心技术、以及Agent应用开发，推荐你学习我刚上线的这门新课：《DeepSeek 系统实战：架构·算法·工程·应用》，这是一套从模型微调、部署，到强化学习训练的系统学习路线，课程以企业级落地为目标，你将掌握LLM核心原理、Agentic RAG、MoE/MLA/MTP机制拆解、PPO/GRPO强化学习与工业级DeepSeek-OCR多模态实战等，想系统掌握并落地这些能力，就从这门课开始。点击以下链接查看课程详情：

B站：<https://www.bilibili.com/cheese/play/ss556613313>

官网（国内访问需科学上网）：<https://www.tgltommy.com/p/deepseek>

© 2026 TGLTommy 原创出品 · All Rights Reserved

B站/公众号/YouTube: 唐国梁Tommy

欢迎关注微信公众号



官方网站: TGLTommy.com(需科学上网)