WebResearchRetriever

Given a query, this retriever will:

- Formulate a set of relate Google searches
- Search for each
- Load all the resulting URLs
- Then embed and perform similarity search with the query on the consolidate page content

```
from langchain.retrievers.web_research import WebResearchRetriever
```

API Reference:

- WebResearchRetriever from `langchain.retrievers.web_research`

Simple usage

Specify the LLM to use for Google search query generation.

```
import os
from langchain.vectorstores import Chroma
from langchain.embeddings import OpenAIEmbeddings
from langchain.chat_models.openai import ChatOpenAI
from langchain.utilities import GoogleSearchAPIWrapper

# Vectorstore
vectorstore =
Chroma(embedding_function=OpenAIEmbeddings(),persist_directory="./chroma_d

# LLM
llm = ChatOpenAI(temperature=0)

# Search
os.environ["GOOGLE_CSE_ID"] = "xxx"
os.environ["GOOGLE_API_KEY"] = "xxx"
search = GoogleSearchAPIWrapper()
```

API Reference:

- Chroma from `langchain.vectorstores`

- OpenAIEmbeddings from `langchain.embeddings`
- ChatOpenAI from `langchain.chat_models.openai`
- GoogleSearchAPIWrapper from `langchain.utilities`

```
# Initialize
web_research_retriever = WebResearchRetriever.from_llm(
    vectorstore=vectorstore,
    llm=llm,
    search=search,
)
```

`Run with citations`

We can use `RetrievalQAWithSourcesChain` to retrieve docs and provide citations

```
from langchain.chains import RetrievalQAWithSourcesChain
user_input = "How do LLM Powered Autonomous Agents work?"
qa_chain =
RetrievalQAWithSourcesChain.from_chain_type(llm,retriever=web_research_ret
result = qa_chain({"question": user_input})
result
```

API Reference:

- RetrievalQAWithSourcesChain from `langchain.chains`

```
    Fetching pages:
100%|######################################################################
1/1 [00:00<00:00,  3.33it/s]





    {'question': 'How do LLM Powered Autonomous Agents work?',
     'answer': "LLM Powered Autonomous Agents work by using LLM (large lar
complemented by several key components, including planning, memory, and to
problem solver. \n",
     'sources': 'https://lilianweng.github.io/posts/2023-06-23-agent/'}
```

`Run with logging`

Here, we use `get_relevant_documents` method to return docs.

```
# Run
import logging
logging.basicConfig()
logging.getLogger("langchain.retrievers.web_research").setLevel(logging.IN
user_input = "What is Task Decomposition in LLM Powered Autonomous Agents?
docs = web_research_retriever.get_relevant_documents(user_input)
```

```
    INFO:langchain.retrievers.web_research:Generating questions for
Google Search ...
    INFO:langchain.retrievers.web_research:Questions for Google Search
(raw): {'question': 'What is Task Decomposition in LLM Powered
Autonomous Agents?', 'text': LineList(lines=['1. How do LLM powered
autonomous agents utilize task decomposition?\n', '2. Can you explain
the concept of task decomposition in LLM powered autonomous agents?\n',
'3. What role does task decomposition play in the functioning of LLM
powered autonomous agents?\n', '4. Why is task decomposition important
for LLM powered autonomous agents?\n'])}
    INFO:langchain.retrievers.web_research:Questions for Google Search:
['1. How do LLM powered autonomous agents utilize task decomposition?
\n', '2. Can you explain the concept of task decomposition in LLM
powered autonomous agents?\n', '3. What role does task decomposition
play in the functioning of LLM powered autonomous agents?\n', '4. Why
is task decomposition important for LLM powered autonomous agents?\n']
    INFO:langchain.retrievers.web_research:Searching for relevat urls
...
    INFO:langchain.retrievers.web_research:Searching for relevat urls
...
    INFO:langchain.retrievers.web_research:Search results: [{'title':
"LLM Powered Autonomous Agents | Lil'Log", 'link':
'https://lilianweng.github.io/posts/2023-06-23-agent/', 'snippet': 'Jun
23, 2023 ... Task decomposition can be done (1) by LLM with simple
prompting like "Steps for XYZ.\\n1." , "What are the subgoals for
achieving XYZ?'}]
    INFO:langchain.retrievers.web_research:Searching for relevat urls
...
    INFO:langchain.retrievers.web_research:Search results: [{'title':
"LLM Powered Autonomous Agents | Lil'Log", 'link':
'https://lilianweng.github.io/posts/2023-06-23-agent/', 'snippet': 'Jun
23, 2023 ... Task decomposition can be done (1) by LLM with simple
prompting like "Steps for XYZ.\\n1." , "What are the subgoals for
achieving XYZ?" , (2)\xa0...'}]
    INFO:langchain.retrievers.web_research:Searching for relevat urls
...
    INFO:langchain.retrievers.web_research:Search results: [{'title':
"LLM Powered Autonomous Agents | Lil'Log", 'link':
'https://lilianweng.github.io/posts/2023-06-23-agent/', 'snippet': 'Jun
23, 2023 ... In a LLM-powered autonomous agent system, LLM functions as
the ... Task decomposition can be done (1) by LLM with simple prompting
like\xa0...'}]
    INFO:langchain.retrievers.web_research:Searching for relevat urls
...
    INFO:langchain.retrievers.web_research:Search results: [{'title':
"LLM Powered Autonomous Agents | Lil'Log", 'link':
'https://lilianweng.github.io/posts/2023-06-23-agent/', 'snippet': 'Jun
23, 2023 ... Agent System Overview In a LLM-powered autonomous agent
system, ... Task decomposition can be done (1) by LLM with simple
prompting like\xa0...'}]
    INFO:langchain.retrievers.web_research:New URLs to load: []
```

`Generate answer using retrieved docs`

We can use `load_qa_chain` for QA using the retrieved docs

```
from langchain.chains.question_answering import load_qa_chain
chain = load_qa_chain(llm, chain_type="stuff")
output = chain({"input_documents": docs, "question":
user_input},return_only_outputs=True)
output['output_text']
```

API Reference:

- load_qa_chain from `langchain.chains.question_answering`

```
    'Task decomposition in LLM-powered autonomous agents refers to the
process of breaking down a complex task into smaller, more manageable
subgoals. This allows the agent to efficiently handle and execute the
individual steps required to complete the overall task. By decomposing
the task, the agent can prioritize and organize its actions, making it
easier to plan and execute the necessary steps towards achieving the
desired outcome.'
```

More flexibility

Pass an LLM chain with custom prompt and output parsing

```
import os
import re
from typing import List
from langchain.chains import LLMChain
from pydantic import BaseModel, Field
from langchain.prompts import PromptTemplate
from langchain.output_parsers.pydantic import PydanticOutputParser

# LLMChain
search_prompt = PromptTemplate(
    input_variables=["question"],
    template="""You are an assistant tasked with improving Google
search
    results. Generate FIVE Google search queries that are similar to
    this question. The output should be a numbered list of questions
and each
    should have a question mark at the end: {question}""",
)

class LineList(BaseModel):
    """List of questions."""

    lines: List[str] = Field(description="Questions")

class QuestionListOutputParser(PydanticOutputParser):
    """Output parser for a list of numbered questions."""

    def __init__(self) -> None:
        super().__init__(pydantic_object=LineList)

    def parse(self, text: str) -> LineList:
        lines = re.findall(r"\d+\..*?\n", text)
        return LineList(lines=lines)

llm_chain = LLMChain(
            llm=llm,
            prompt=search_prompt,
            output_parser=QuestionListOutputParser(),
        )
```

API Reference:

- LLMChain from `langchain.chains`
- PromptTemplate from `langchain.prompts`
- PydanticOutputParser from `langchain.output_parsers.pydantic`

```
# Initialize
web_research_retriever_llm_chain = WebResearchRetriever(
    vectorstore=vectorstore,
    llm_chain=llm_chain,
    search=search,
)

# Run
docs =
web_research_retriever_llm_chain.get_relevant_documents(user_input)
```

```
    INFO:langchain.retrievers.web_research:Generating questions for Google
    INFO:langchain.retrievers.web_research:Questions for Google Search (ra
Autonomous Agents?', 'text': LineList(lines=['1. How do LLM powered autono
decomposition important for LLM powered autonomous agents?\n', '3. Can you
autonomous agents?\n', '4. What are the benefits of task decomposition in
    INFO:langchain.retrievers.web_research:Questions for Google Search: ['
\n', '2. Why is task decomposition important for LLM powered autonomous ag
in LLM powered autonomous agents?\n', '4. What are the benefits of task de
    INFO:langchain.retrievers.web_research:Searching for relevat urls ...
    INFO:langchain.retrievers.web_research:Searching for relevat urls ...
    INFO:langchain.retrievers.web_research:Search results: [{'title': "LLM
'https://lilianweng.github.io/posts/2023-06-23-agent/', 'snippet': 'Jun 23
simple prompting like "Steps for XYZ.\\n1." , "What are the subgoals for a
    INFO:langchain.retrievers.web_research:Searching for relevat urls ...
    INFO:langchain.retrievers.web_research:Search results: [{'title': "LLM
'https://lilianweng.github.io/posts/2023-06-23-agent/', 'snippet': 'Jun 23
simple prompting like "Steps for XYZ.\\n1." , "What are the subgoals for a
    INFO:langchain.retrievers.web_research:Searching for relevat urls ...
    INFO:langchain.retrievers.web_research:Search results: [{'title': "LLM
'https://lilianweng.github.io/posts/2023-06-23-agent/', 'snippet': 'Jun 23
simple prompting like "Steps for XYZ.\\n1." , "What are the subgoals for a
    INFO:langchain.retrievers.web_research:Searching for relevat urls ...
    INFO:langchain.retrievers.web_research:Search results: [{'title': "LLM
'https://lilianweng.github.io/posts/2023-06-23-agent/', 'snippet': 'Jun 23
simple prompting like "Steps for XYZ.\\n1." , "What are the subgoals for a
    INFO:langchain.retrievers.web_research:New URLs to load: ['https://lil
    INFO:langchain.retrievers.web_research:Grabbing most relevant splits f
    Fetching pages:
100%|##########################################################################
1/1 [00:00<00:00,  6.32it/s]
```

```
len(docs)
```

```
    1
```

Run locally

Specify LLM and embeddings that will run locally (e.g., on your laptop)

```
from langchain.llms import LlamaCpp
from langchain.embeddings import GPT4AllEmbeddings
from langchain.callbacks.manager import CallbackManager
from langchain.callbacks.streaming_stdout import StreamingStdOutCallbackHa

n_gpu_layers = 1  # Metal set to 1 is enough.
n_batch = 512  # Should be between 1 and n_ctx, consider the amount of RAM
Apple Silicon Chip.
callback_manager = CallbackManager([StreamingStdOutCallbackHandler()])
llama = LlamaCpp(
    model_path="/Users/rlm/Desktop/Code/llama.cpp/llama-2-13b-chat.ggmlv3.
    n_gpu_layers=n_gpu_layers,
    n_batch=n_batch,
    n_ctx=4096,  # Context window
    max_tokens=1000,  # Max tokens to generate
    f16_kv=True,  # MUST set to True, otherwise you will run into problem
couple of calls
    callback_manager=callback_manager,
    verbose=True,
)

vectorstore_llama =
Chroma(embedding_function=GPT4AllEmbeddings(),persist_directory="./chroma_
```

API Reference:

- LlamaCpp from `langchain.llms`
- GPT4AllEmbeddings from `langchain.embeddings`
- CallbackManager from `langchain.callbacks.manager`
- StreamingStdOutCallbackHandler from `langchain.callbacks.streaming_stdout`

```
    llama.cpp: loading model from
/Users/rlm/Desktop/Code/llama.cpp/llama-2-13b-chat.ggmlv3.q4_0.bin
    llama_model_load_internal: format     = ggjt v3 (latest)
    llama_model_load_internal: n_vocab    = 32000
    llama_model_load_internal: n_ctx      = 4096
    llama_model_load_internal: n_embd     = 5120
    llama_model_load_internal: n_mult     = 256
    llama_model_load_internal: n_head     = 40
    llama_model_load_internal: n_layer    = 40
    llama_model_load_internal: n_rot      = 128
    llama_model_load_internal: freq_base  = 10000.0
    llama_model_load_internal: freq_scale = 1
    llama_model_load_internal: ftype      = 2 (mostly Q4_0)
    llama_model_load_internal: n_ff       = 13824
    llama_model_load_internal: model size = 13B
    llama_model_load_internal: ggml ctx size =    0.09 MB
    llama_model_load_internal: mem required  = 9132.71 MB (+ 1608.00 MB
per state)
    llama_new_context_with_model: kv self size  = 3200.00 MB
    ggml_metal_init: allocating


    Found model file at  /Users/rlm/.cache/gpt4all/ggml-all-MiniLM-L6-
v2-f16.bin
    llama_new_context_with_model: max tensor size =    87.89 MB


    ggml_metal_init: using MPS
    ggml_metal_init: loading
'/Users/rlm/miniforge3/envs/llama/lib/python3.9/site-
packages/llama_cpp/ggml-metal.metal'
    ggml_metal_init: loaded kernel_add
0x110fbd600
    ggml_metal_init: loaded kernel_mul
0x110fbeb30
    ggml_metal_init: loaded kernel_mul_row
0x110fbf350
    ggml_metal_init: loaded kernel_scale
0x110fbf9e0
    ggml_metal_init: loaded kernel_silu
0x110fc0150
    ggml_metal_init: loaded kernel_relu
0x110fbd950
    ggml_metal_init: loaded kernel_gelu
0x110fbdbb0
    ggml_metal_init: loaded kernel_soft_max
0x110fc14d0
    ggml_metal_init: loaded kernel_diag_mask_inf
0x110fc1980
    ggml_metal_init: loaded kernel_get_rows_f16
0x110fc22a0
    ggml_metal_init: loaded kernel_get_rows_q4_0
```

```
0x110fc2ad0
    ggml_metal_init: loaded kernel_get_rows_q4_1
0x110fc3260
    ggml_metal_init: loaded kernel_get_rows_q2_K
0x110fc3ad0
    ggml_metal_init: loaded kernel_get_rows_q3_K
0x110fc41c0
    ggml_metal_init: loaded kernel_get_rows_q4_K
0x110fc48c0
    ggml_metal_init: loaded kernel_get_rows_q5_K
0x110fc4fa0
    ggml_metal_init: loaded kernel_get_rows_q6_K
0x110fc56a0
    ggml_metal_init: loaded kernel_rms_norm
0x110fc5da0
    ggml_metal_init: loaded kernel_norm
0x110fc64d0
    ggml_metal_init: loaded kernel_mul_mat_f16_f32
0x2a5c19990
    ggml_metal_init: loaded kernel_mul_mat_q4_0_f32
0x2a5c1d4a0
    ggml_metal_init: loaded kernel_mul_mat_q4_1_f32
0x2a5c19fc0
    ggml_metal_init: loaded kernel_mul_mat_q2_K_f32
0x2a5c1dcc0
    ggml_metal_init: loaded kernel_mul_mat_q3_K_f32
0x2a5c1e420
    ggml_metal_init: loaded kernel_mul_mat_q4_K_f32
0x2a5c1edc0
    ggml_metal_init: loaded kernel_mul_mat_q5_K_f32
0x2a5c1fd90
    ggml_metal_init: loaded kernel_mul_mat_q6_K_f32
0x2a5c20540
    ggml_metal_init: loaded kernel_rope
0x2a5c20d40
    ggml_metal_init: loaded kernel_alibi_f32
0x2a5c21730
    ggml_metal_init: loaded kernel_cpy_f32_f16
0x2a5c21ab0
    ggml_metal_init: loaded kernel_cpy_f32_f32
0x2a5c22080
    ggml_metal_init: loaded kernel_cpy_f16_f16
0x2a5c231d0
    ggml_metal_init: recommendedMaxWorkingSetSize = 21845.34 MB
    ggml_metal_init: hasUnifiedMemory             = true
    ggml_metal_init: maxTransferRate              = built-in GPU
    ggml_metal_add_buffer: allocated 'data            ' buffer, size =
6984.06 MB, ( 6984.52 / 21845.34)
    ggml_metal_add_buffer: allocated 'eval            ' buffer, size =
1040.00 MB, ( 8024.52 / 21845.34)
    ggml_metal_add_buffer: allocated 'kv              ' buffer, size =
3202.00 MB, (11226.52 / 21845.34)
    ggml_metal_add_buffer: allocated 'scr0            ' buffer, size =
597.00 MB, (11823.52 / 21845.34)
    AVX = 0 | AVX2 = 0 | AVX512 = 0 | AVX512_VBMI = 0 | AVX512_VNNI = 0
| FMA = 0 | NEON = 1 | ARM_FMA = 1 | F16C = 0 | FP16_VA = 1 | WASM_SIMD
= 0 | BLAS = 1 | SSE3 = 0 | VSX = 0 |
    ggml_metal_add_buffer: allocated 'scr1            ' buffer, size =
512.00 MB, (12335.52 / 21845.34)
    objc[33471]: Class GGMLMetalClass is implemented in both
/Users/rlm/miniforge3/envs/llama/lib/python3.9/site-
packages/llama_cpp/libllama.dylib (0x2c7368208) and
/Users/rlm/miniforge3/envs/llama/lib/python3.9/site-
packages/gpt4all/llmodel_DO_NOT_MODIFY/build/libreplit-mainline-
metal.dylib (0x5ebf48208). One of the two will be used. Which one is
undefined.
    objc[33471]: Class GGMLMetalClass is implemented in both
/Users/rlm/miniforge3/envs/llama/lib/python3.9/site-
packages/llama_cpp/libllama.dylib (0x2c7368208) and
/Users/rlm/miniforge3/envs/llama/lib/python3.9/site-
packages/gpt4all/llmodel_DO_NOT_MODIFY/build/libllamamodel-mainline-
metal.dylib (0x5ec374208). One of the two will be used. Which one is
undefined.
```

We supplied `StreamingStdOutCallbackHandler()`, so model outputs (e.g., generated questions) are streamed.

We also have logging on, so we seem them there too.

```
from langchain.chains import RetrievalQAWithSourcesChain
# Initialize
web_research_retriever = WebResearchRetriever.from_llm(
    vectorstore=vectorstore_llama,
    llm=llama,
    search=search,
)

# Run
user_input = "What is Task Decomposition in LLM Powered Autonomous Agents
qa_chain =
RetrievalQAWithSourcesChain.from_chain_type(llama,retriever=web_research_
result = qa_chain({"question": user_input})
result
```

API Reference:

- RetrievalQAWithSourcesChain from `langchain.chains`

```
    INFO:langchain.retrievers.web_research:Generating questions for Google


      Sure, here are five Google search queries that are similar to "What

    1. How does Task Decomposition work in LLM Powered Autonomous Agents?
    2. What are the benefits of using Task Decomposition in LLM Powered Au
    3. Can you provide examples of Task Decomposition in LLM Powered Autor
    4. How does Task Decomposition improve the performance of LLM Powered
    5. What are some common challenges or limitations of using Task Decomp
addressed?



    llama_print_timings:        load time =  8585.01 ms
    llama_print_timings:      sample time =   124.24 ms /   164 runs   (
    llama_print_timings: prompt eval time =  8584.83 ms /   101 tokens (
    llama_print_timings:        eval time =  7268.55 ms /   163 runs   (
    llama_print_timings:       total time = 16236.13 ms
    INFO:langchain.retrievers.web_research:Questions for Google Search (ra
Autonomous Agents?', 'text': LineList(lines=['1. How does Task Decomposit
benefits of using Task Decomposition in LLM Powered Autonomous Agents? \n
Powered Autonomous Agents? \n', '4. How does Task Decomposition improve th
    INFO:langchain.retrievers.web_research:Questions for Google Search: [
Agents? \n', '2. What are the benefits of using Task Decomposition in LLM
Task Decomposition in LLM Powered Autonomous Agents? \n', '4. How does Tas
Autonomous Agents? \n']
    INFO:langchain.retrievers.web_research:Searching for relevat urls ...
    INFO:langchain.retrievers.web_research:Searching for relevat urls ...
    INFO:langchain.retrievers.web_research:Search results: [{'title': "LLM
'https://lilianweng.github.io/posts/2023-06-23-agent/', 'snippet': 'Jun 23
simple prompting like "Steps for XYZ.\\n1." , "What are the subgoals for a
    INFO:langchain.retrievers.web_research:Searching for relevat urls ...
    INFO:langchain.retrievers.web_research:Search results: [{'title': "LLM
'https://lilianweng.github.io/posts/2023-06-23-agent/', 'snippet': 'Jun 23
simple prompting like "Steps for XYZ.\\n1." , "What are the subgoals for a
    INFO:langchain.retrievers.web_research:Searching for relevat urls ...
    INFO:langchain.retrievers.web_research:Search results: [{'title': "LLM
'https://lilianweng.github.io/posts/2023-06-23-agent/', 'snippet': 'Jun 23
agent needs to know what they are and plan ahead. Task Decomposition#. Cha
```

```
    INFO:langchain.retrievers.web_research:Searching for relevat urls ...
    INFO:langchain.retrievers.web_research:Search results: [{'title': "LLM
'https://lilianweng.github.io/posts/2023-06-23-agent/', 'snippet': 'Jun 23
agent system, ... Task decomposition can be done (1) by LLM with simple pr
    INFO:langchain.retrievers.web_research:New URLs to load: ['https://lil
    INFO:langchain.retrievers.web_research:Grabbing most relevant splits f
    Fetching pages:
100%|##########################################################################
1/1 [00:00<00:00, 10.49it/s]
    Llama.generate: prefix-match hit


     The content discusses Task Decomposition in LLM Powered Autonomous Ag
manageable subgoals for efficient handling of complex tasks.
    SOURCES:
    https://lilianweng.github.io/posts/2023-06-23-agent/



    llama_print_timings:        load time =  8585.01 ms
    llama_print_timings:      sample time =    52.88 ms /    72 runs   (
    llama_print_timings: prompt eval time = 125925.13 ms /  2358 tokens (
    llama_print_timings:        eval time =  3504.16 ms /    71 runs   (
    llama_print_timings:       total time = 129584.60 ms





    {'question': 'What is Task Decomposition in LLM Powered Autonomous Age
     'answer': ' The content discusses Task Decomposition in LLM Powered A
smaller, manageable subgoals for efficient handling of complex tasks.\n',
     'sources': 'https://lilianweng.github.io/posts/2023-06-23-agent/'}
```