LangChain Document Loader

How to use the LangChain document loader to ingest Stack Overflow data into an LLM.

Document generated 06/30/2025

PDF VERSION

Tags | **API** | **AI** | **LangChain** |

Applies to: Free, Basic, Business | Enterprise

Overview

A LangChain document loader uses the LangChain framework to simplify the ingestion of data into an LLM (large language model). The langchain-stack-overflow-for-teams document loader allows you to load your Teams Basic, Business, or Enterprise content into a variety of LLMs and frameworks that support the LangChain format. Learn more about the LangChain framework.

NOTE: *The code and examples provided are intended for demonstration purposes only. While we strive to ensure accuracy and clarity, the code is provided "as-is" without guarantees, warranties, or support of any kind. For any production use, we recommend reviewing the code thoroughly and adapting it to meet your specific requirements and environment. Use of this material is at your own discretion and risk.*

Installation

The Stack Overflow for Teams LangChain document loader is a Python module you can access here: langchain-stack-overflow-for-teams (StackOverflowTeamsApiV3Loader). To install the module, use a Python package management solution such as pip or uv. For example:

- `pip install langchain-stack-overflow-for-teams`
- `uv add langchain-stack-overflow-for-teams`

Supported parameters

The StackOverflowTeamsApiV3Loader supports a number of parameters to allow flexibility in retrieving data from your site.

Parameter	Required	Description
`access_token`	Yes	The API v3 access token for your Stack Overflow for Teams site.
`endpoint`	Teams Basic, Business: No Teams Enterprise: Yes	Your site's API v3 endpoint. This defaults to api.stackoverflowteams.com for Teams Basic and Business. Teams Enterprise users must set this parameter (for example: [your_site].stackenterprise.co/api).

Parameter	Required	Description
`team`	Teams Basic, Business: Yes Teams Enterprise: No	The team to retrieve data from. This is required for Teams Basic and Business. Set this for Teams Enterprise to retrieve data from a private team (optional).
`content_type`	Yes	The content type to retrieve: questions (and corresponding answers) or articles.
`date_from`	No	The data retrieval start date in ISO 8601 format (YYYY-MM-DDThh:mm:ssZ) format, useful for returning only the newest data.
`sort`	No	The attribute to sort results by: "activity", "creation", or "score".
`order`	No	The order to sort results by: "asc" or “desc”.
`is_answered`	No	Return answered questions that have at least one positively-scored answer (doesn't apply to article content type): "true" or "false"
`has_accepted_answer`	No	Return questions with *accepted* answers only (doesn't apply to article content type): "true" or "false"
`max_retries`	No	The number of retries for the request: number (default is 3)
`timeout`	No	The timeout for the request: number (default is 30 seconds)

Authentication

The LangChain loader authenticates using an API v3 auth token previously created in Stack Overflow for Teams. The process for creating a token differs for Stack Overflow for Teams Basic, Business, and Enterprise sites.

Stack Overflow for Teams Basic and Business use a **personal access token (PAT)**. Learn more about PAT authentication in this Stack Overflow for Teams Basic and Business API v3 Overview article.

Stack Overflow for Teams Enterprise uses a **service key access token**. Learn more about service keys and access tokens in this Stack Overflow for Teams Enterprise API v3 Overview article.

Examples

These examples demonstrate how to use the library to authenticate into and retrieve data from a Stack Overflow for Teams site. The last example demonstrates a fully functional implementation of the LangChain loader and LanceDB vector store, one of many possible methods to ingest your data into an LLM.

Get articles from Teams Basic or Business

```
from langchain_stack_overflow_for_teams import StackOverflowTeamsApiV3Loader

loader = StackOverflowTeamsApiV3Loader(
   access_token=os.environ.get("SO_PAT"),
   team="my team",
   content_type="articles",
```

```
)
docs = loader.load()
```

Get questions with accepted answers from Teams Basic or Business

```
from langchain_stack_overflow_for_teams import StackOverflowTeamsApiV3Loader

loader = StackOverflowTeamsApiV3Loader(
   access_token=os.environ.get("SO_PAT"),
   team="my team",
   content_type="questions",
   has_accepted_answer="true",
,
)
docs = loader.load()
```

Get articles from Teams Enterprise

Unlike the previous examples that use the default Teams Basic and Business endpoint, the following Teams Enterprise examples specify the site URL.

```
from langchain_stack_overflow_for_teams import StackOverflowTeamsApiV3Loader

loader = StackOverflowTeamsApiV3Loader(
   endpoint="[your_site].stackenterprise.co/api",
   access_token=os.environ.get("SO_API_TOKEN"),
   content_type="articles",
)
docs = loader.load()
```

Get articles from a private team in Teams Enterprise

```
from langchain_stack_overflow_for_teams import StackOverflowTeamsApiV3Loader

loader = StackOverflowTeamsApiV3Loader(
   endpoint="[your_site].stackenterprise.co/api",
   access_token=os.environ.get("SO_API_TOKEN"),
   team="my team",
   content_type="articles",
)
docs = loader.load()
```

Full Example

This example retrieves content from a Stack Overflow for Teams Enterprise site and loads it into a LanceDB vector store for access by an LLM-based system.

This example uses all available parameters to retrieve questions:

- with at least one positively-scored answer
- with at least one accepted answer

- from a private team
- since a specified date
- sorted by activity, descending

```
""" This script demonstrates use the Langchain add_documents model to naively load all documents every time (easy, but no
import os
from dotenv import load_dotenv
from langchain_openai import AzureChatOpenAI, AzureOpenAIEmbeddings
from langchain_community.vectorstores import LanceDB
from langchain_text_splitters import HTMLSemanticPreservingSplitter
from langchain_stack_overflow_for_teams import StackOverflowTeamsApiV3Loader
from langchain.tools.retriever import create_retriever_tool
from langchain_core.prompts import ChatPromptTemplate
from langchain.agents import AgentExecutor, create_tool_calling_agent

def main():
   load_dotenv()

   # initialize the LLM using gpt-4o-mini from Azure OpenAI
   llm = AzureChatOpenAI(
       azure_deployment="gpt-4o-mini",
       api_version="2025-01-01-preview",
   )

   # initialize the embeddings using Azure OpenAI
   embeddings = AzureOpenAIEmbeddings()

   # initialize the LanceDB vector store
   db = LanceDB(
       table_name="docs",
       uri="./db/lancedb",
       embedding=embeddings,
   )

   # load documents using our StackOverflowTeamsApiV3Loader document loader
   loader = StackOverflowTeamsApiV3Loader(
```

Special thanks to Stack Overflow's Ray Terrill for the content of this article.

Need help? Submit an issue or question through our support portal.