SearxNG Search API

Contents

This notebook goes over how to use a self hosted SearxNG search API to search the web.

You can check this link for more informations about Searx API parameters.

```
import pprint
from langchain.utilities import SearxSearchWrapper
```

```
search = SearxSearchWrapper(searx_host="http://127.0.0.1:8888")
```

For some engines, if a direct `answer` is available the warpper will print the answer instead of the full list of search results. You can use the `results` method of the wrapper if you want to obtain all the results.

```
search.run("What is the capital of France")
```

```
'Paris is the capital of France, the largest country of Europe with 550 000 km2
(65 millions inhabitants). Paris has 2.234 million inhabitants end 2011. She is
the core of Ile de France region (12 million people).'
```

Custom Parameters

SearxNG supports up to 139 search engines. You can also customize the Searx wrapper with arbitrary named parameters that will be passed to the Searx search API . In the below example

Skip to main content

In this example we will be using the `engines` parameters to query wikipedia

```
search = SearxSearchWrapper(searx_host="http://127.0.0.1:8888", k=5) # k is for
max number of items
```

```
search.run("large language model ", engines=['wiki'])
```

```
'Large language models (LLMs) represent a major advancement in AI, with the
promise of transforming domains through learned knowledge. LLM sizes have been
increasing 10X every year for the last few years, and as these models grow in
complexity and size, so do their capabilities.\n\nGPT-3 can translate language,
write essays, generate computer code, and more — all with limited to no
supervision. In July 2020, OpenAI unveiled GPT-3, a language model that was easily
the largest known at the time. Put simply, GPT-3 is trained to predict the next
word in a sentence, much like how a text message autocomplete feature works.\n\nA
large language model, or LLM, is a deep learning algorithm that can recognize,
summarize, translate, predict and generate text and other content based on
knowledge gained from massive datasets. Large language models are among the most
successful applications of transformer models.\n\nAll of today's well-known
language models—e.g., GPT-3 from OpenAI, PaLM or LaMDA from Google, Galactica or
OPT from Meta, Megatron-Turing from Nvidia/Microsoft, Jurassic-1 from AI21 Labs—
are...\n\nLarge language models (LLMs) such as GPT-3are increasingly being used to
generate text. These tools should be used with care, since they can generate
content that is biased, non-verifiable, constitutes original research, or violates
copyrights.'
```

Passing other Searx parameters for searx like `language`

```
search = SearxSearchWrapper(searx_host="http://127.0.0.1:8888", k=1)
search.run("deep learning", language='es', engines=['wiki'])
```

```
'Aprendizaje profundo (en inglés, deep learning) es un conjunto de algoritmos de
aprendizaje automático (en inglés, machine learning) que intenta modelar
abstracciones de alto nivel en datos usando arquitecturas computacionales que
admiten transformaciones no lineales múltiples e iterativas de datos expresados en
forma matricial o tensorial. 1'
```

Obtaining results with metadata

Skip to main content

We also would like to obtain the results in a structured way including metadata. For this we will be using the `results` method of the wrapper.

```
search = SearxSearchWrapper(searx_host="http://127.0.0.1:8888")
```

```
results = search.results("Large Language Model prompt", num_results=5,
categories='science', time_range='year')
pprint.pp(results)
```

```
[{'snippet': '… on natural language instructions, large language models (… the '
             'prompt used to steer the model, and most effective prompts … to '
             'prompt engineering, we propose Automatic Prompt …',
  'title': 'Large language models are human-level prompt engineers',
  'link': 'https://arxiv.org/abs/2211.01910',
  'engines': ['google scholar'],
  'category': 'science'},
 {'snippet': '… Large language models (LLMs) have introduced new possibilities '
             'for prototyping with AI [18]. Pre-trained on a large amount of '
             'text data, models … language instructions called prompts. …',
  'title': 'Promptchainer: Chaining large language model prompts through '
           'visual programming',
  'link': 'https://dl.acm.org/doi/abs/10.1145/3491101.3519729',
  'engines': ['google scholar'],
  'category': 'science'},
 {'snippet': '… can introspect the large prompt model. We derive the view '
             'φ0(X) and the model h0 from T01. However, instead of fully '
             'fine-tuning T0 during co-training, we focus on soft prompt '
             'tuning, …',
  'title': 'Co-training improves prompt-based learning for large language '
```

Get papers from arxiv

```
results = search.results("Large Language Model prompt", num_results=5, engines=
['arxiv'])
pprint.pp(results)
```

```
[{'snippet': 'Thanks to the advanced improvement of large pre-trained language '
             'models, prompt-based fine-tuning is shown to be effective on a '
             'variety of downstream tasks. Though many prompting methods have '
             'been investigated, it remains unknown which type of prompts are '
             'the most effective among three types of prompts (i.e., '
             'human-designed prompts, schema prompts and null prompts). In '
             'this work, we empirically compare the three types of prompts '
```

Skip to main content

```
             'effective in general. Besides, the performance gaps tend to '
             'diminish when the scale of training data grows large.',
  'title': 'Do Prompts Solve NLP Tasks Using Natural Language?',
  'link': 'http://arxiv.org/abs/2203.00902v1',
  'engines': ['arxiv'],
  'category': 'science'},
 {'snippet': 'Cross-prompt automated essay scoring (AES) requires the system '
             'to use non target-prompt essays to award scores to a '
             'target-prompt essay. Since obtaining a large quantity of '
             'pre-graded essays to a particular prompt is often difficult and '
             'unrealistic, the task of cross-prompt AES is vital for the '
```

In this example we query for `large language models` under the `it` category. We then filter the results that come from github.

```
results = search.results("large language model", num_results = 20, categories='it')
pprint.pp(list(filter(lambda r: r['engines'][0] == 'github', results)))
```

```
[{'snippet': 'Guide to using pre-trained large language models of source code',
  'title': 'Code-LMs',
  'link': 'https://github.com/VHellendoorn/Code-LMs',
  'engines': ['github'],
  'category': 'it'},
 {'snippet': 'Dramatron uses large language models to generate coherent '
             'scripts and screenplays.',
  'title': 'dramatron',
  'link': 'https://github.com/deepmind/dramatron',
  'engines': ['github'],
  'category': 'it'}]
```

We could also directly query for results from `github` and other source forges.

```
results = search.results("large language model", num_results = 20, engines=
['github', 'gitlab'])
pprint.pp(results)
```

```
[{'snippet': "Implementation of 'A Watermark for Large Language Models' paper "
             'by Kirchenbauer & Geiping et. al.',
  'title': 'Peutlefaire / LMWatermark',
  'link': 'https://gitlab.com/BrianPulfer/LMWatermark',
  'engines': ['gitlab'],
  'category': 'it'},
 {'snippet': 'Guide to using pre-trained large language models of source code',
  'title': 'Code-LMs',
  'link': 'https://github.com/VHellendoorn/Code-LMs'
```

Skip to main content

```
{'snippet': '',
 'title': 'Simen Burud / Large-scale Language Models for Conversational '
          'Speech Recognition',
 'link': 'https://gitlab.com/BrianPulfer',
 'engines': ['gitlab'],
 'category': 'it'},
{'snippet': 'Dramatron uses large language models to generate coherent '
            'scripts and screenplays.',
 'title': 'dramatron',
```