Who killed Laura Palmer?

How to implement a **Question Answering system** based on a TV series wiki

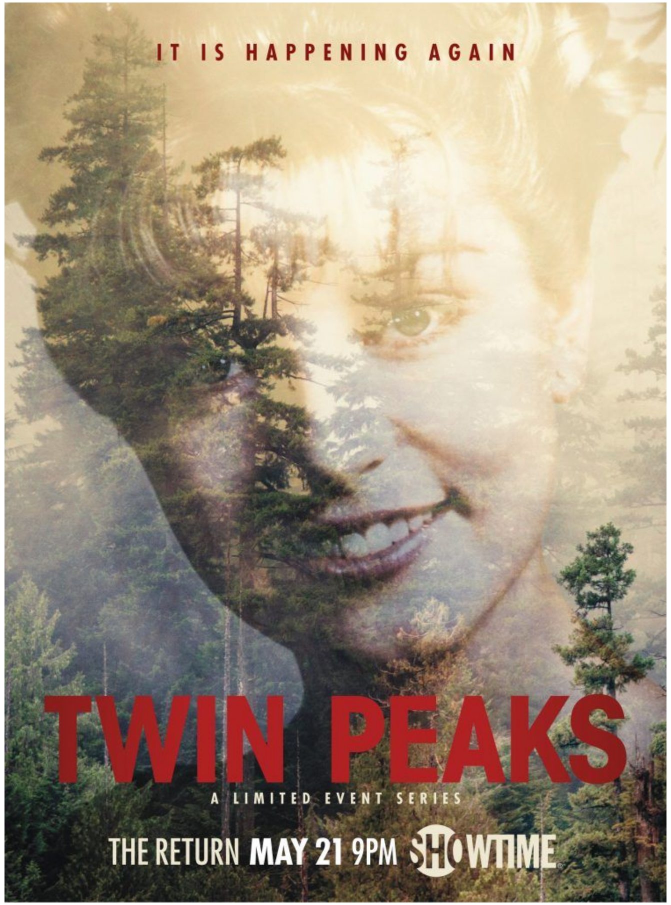
Stefano Fiorucci

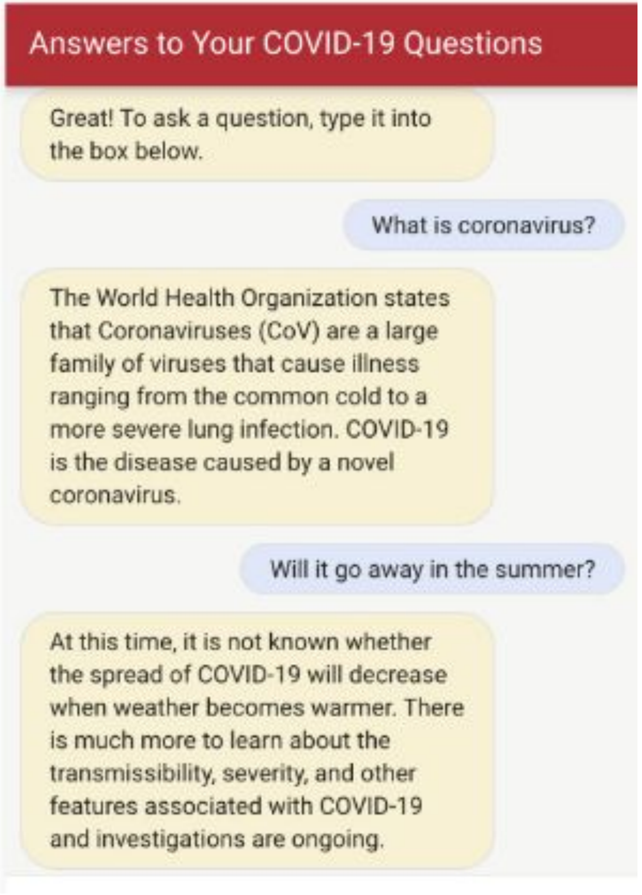
Stefano Fiorucci

- Machine Learning engineer
- NLP enthusiast
- ❤️ Python…

Find me on:
Github (@anakin87) - Linkedin

…during the pandemic…




Answers to Your COVID-19 Questions

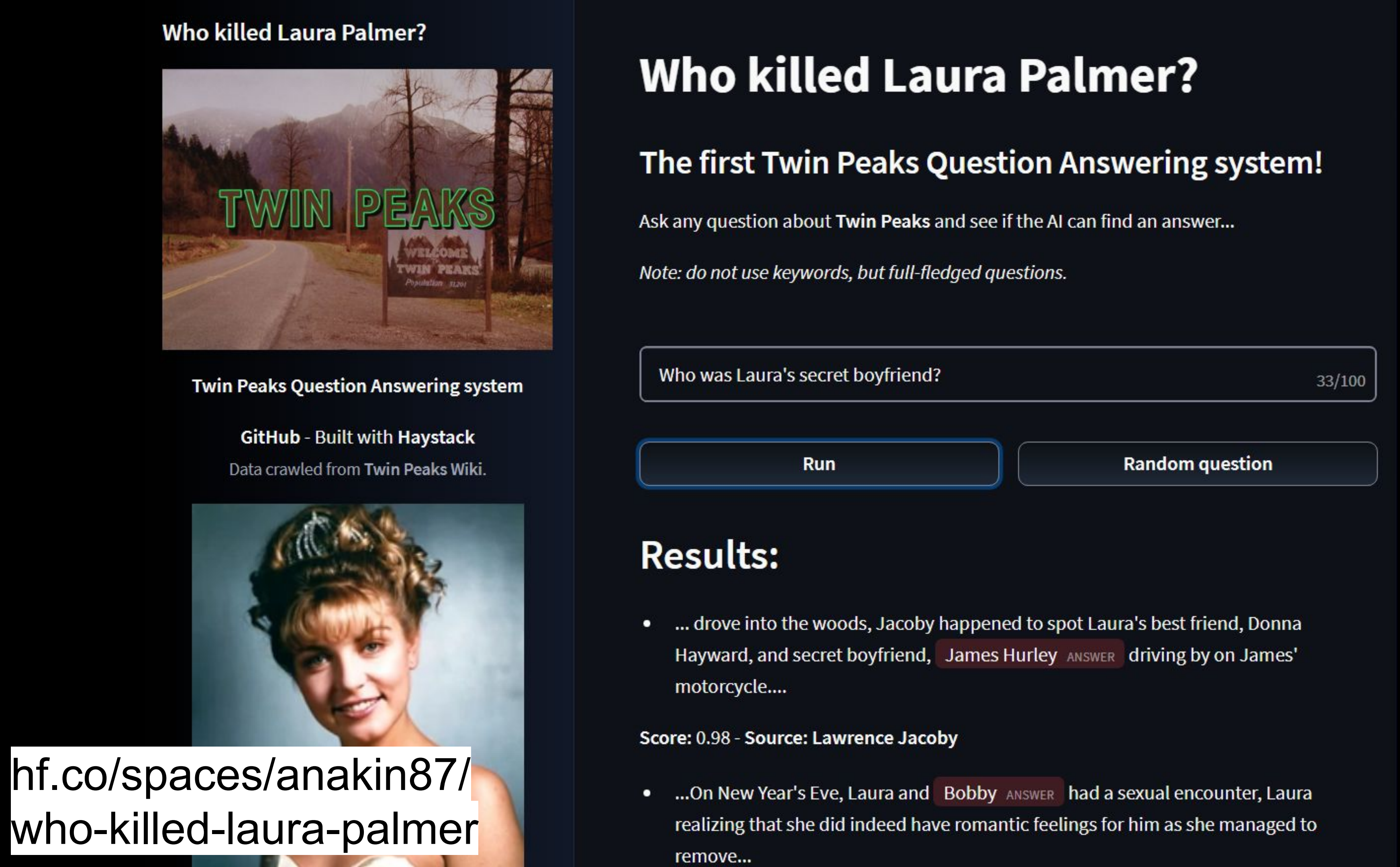
Great! To ask a question, type it into the box below.

What is coronavirus?

The World Health Organization states that Coronaviruses (CoV) are a large family of viruses that cause illness ranging from the common cold to a more severe lung infection. COVID-19 is the disease caused by a novel coronavirus.

Will it go away in the summer?

At this time, it is not known whether the spread of COVID-19 will decrease when weather becomes warmer. There is much more to learn about the transmissibility, severity, and other features associated with COVID-19 and investigations are ongoing.



Who killed Laura Palmer?

Twin Peaks Question Answering system

GitHub - Built with **Haystack**

Data crawled from **Twin Peaks Wiki**.

Who killed Laura Palmer?

The first Twin Peaks Question Answering system!

Ask any question about **Twin Peaks** and see if the AI can find an answer...

Note: do not use keywords, but full-fledged questions.

Who was Laura's secret boyfriend? 33/100

Run

Random question

Results:

- ... drove into the woods, Jacoby happened to spot Laura's best friend, Donna Hayward, and secret boyfriend, James Hurley ANSWER driving by on James' motorcycle....

Score: 0.98 - **Source: Lawrence Jacoby**

- ...On New Year's Eve, Laura and Bobby ANSWER had a sexual encounter, Laura realizing that she did indeed have romantic feelings for him as she managed to remove...

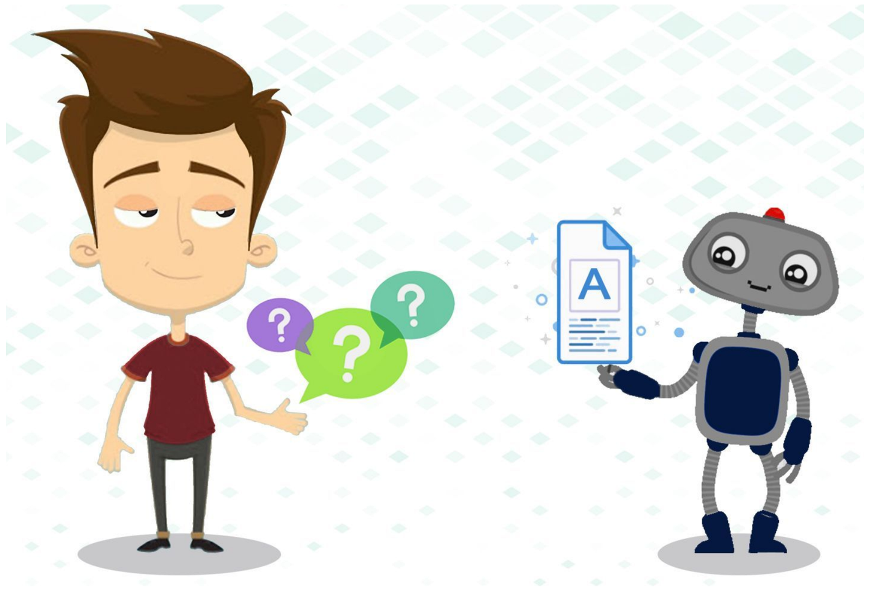
hf.co/spaces/anakin87/who-killed-laura-palmer

Agenda

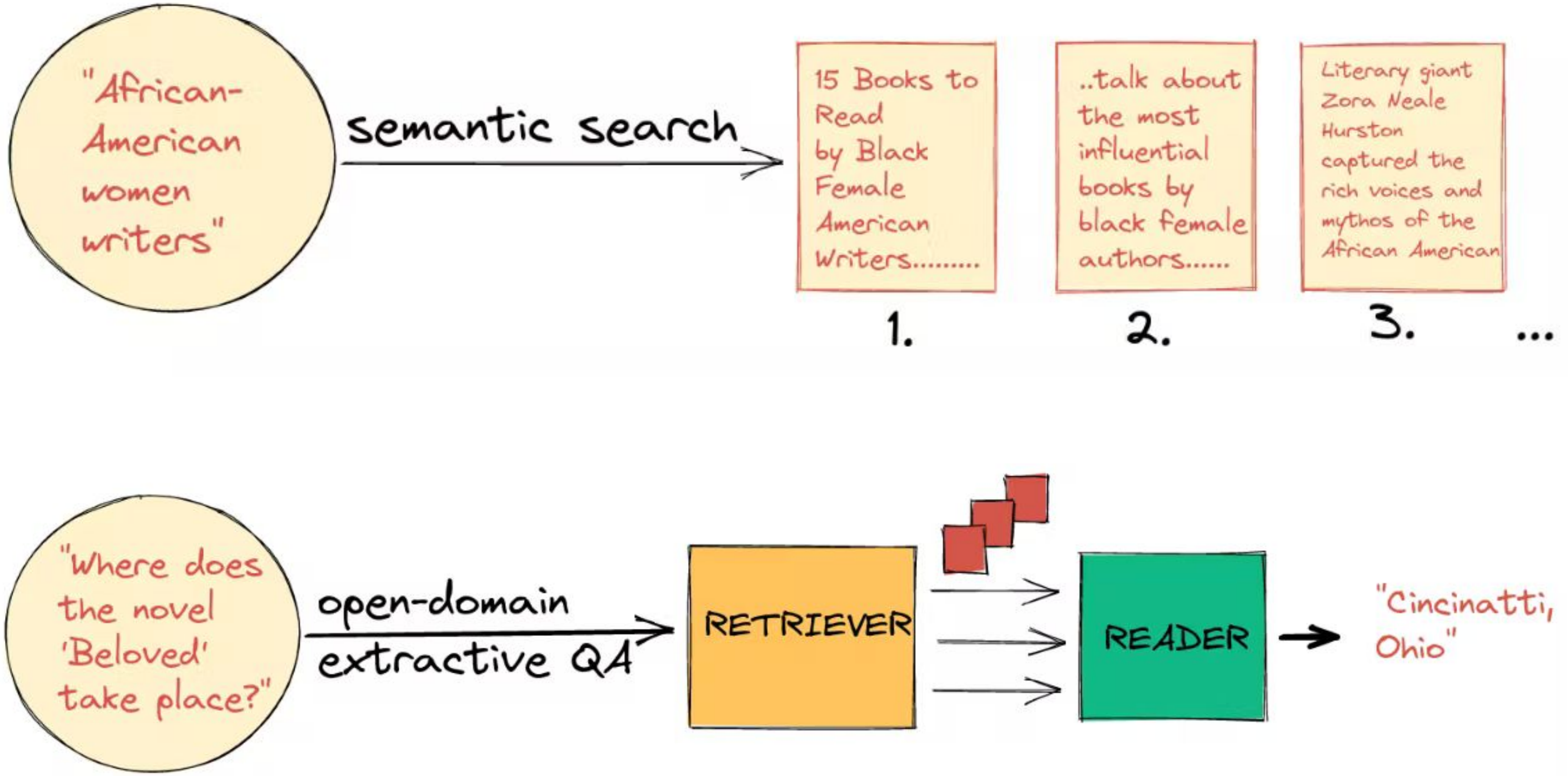
- Theoretical foundations
  - Basics of Question Answering
  - QA architecture
  - Retrieval: sparse vs dense text representations
  - Vector search in a nutshell
  - The Reader
- Practical QA with *Haystack*
  - main features
  - working examples
  - other use cases

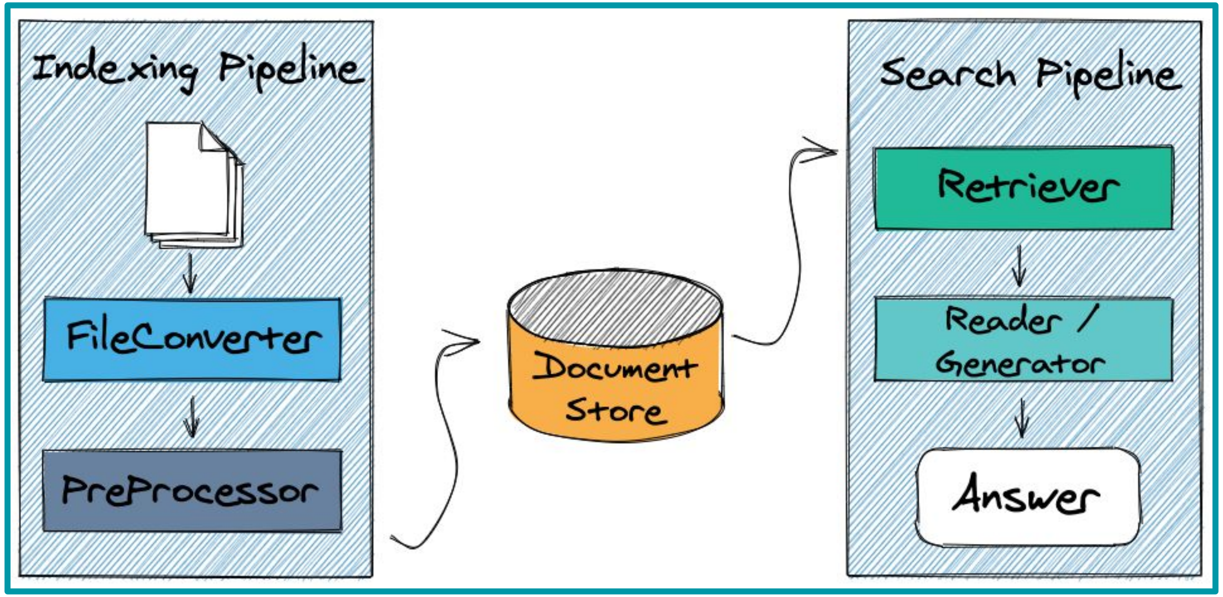
Question Answering: definitions

From Wikipedia: Question answering (QA) is a computer science discipline within the fields of information retrieval and natural language processing (NLP), which is concerned with building systems that **automatically answer questions posed by humans in a natural language**.



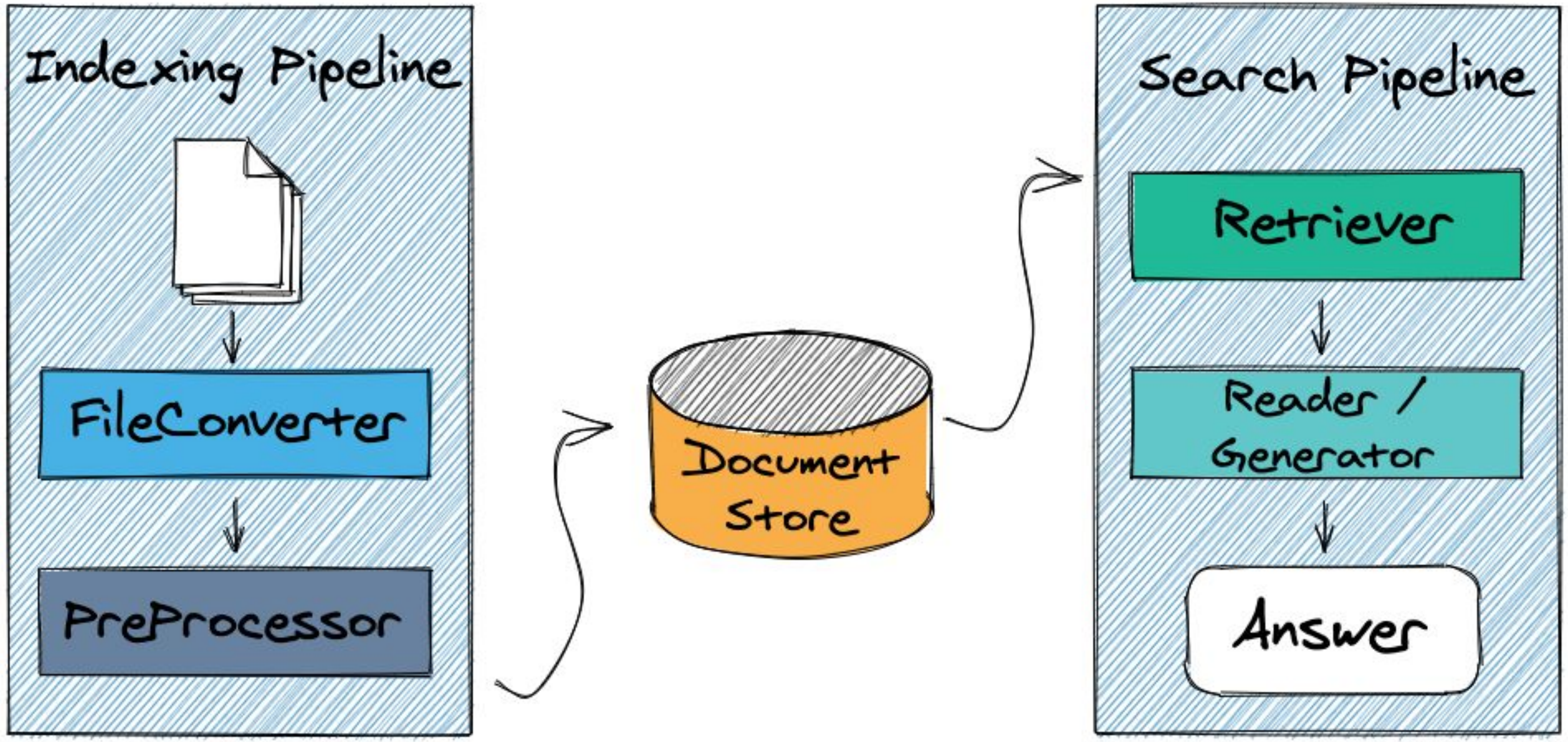
Semantic search vs Question Answering


"African-American women writers"
semantic search
15 Books to Read by Black Female American Writers.........
..talk about the most influential books by black female authors......
Literary giant Zora Neale Hurston captured the rich voices and mythos of the African American
1.
2.
3.
...
"Where does the novel 'Beloved' take place?"
open-domain
extractive QA
RETRIEVER
READER
"Cincinatti, Ohio"



Project overview

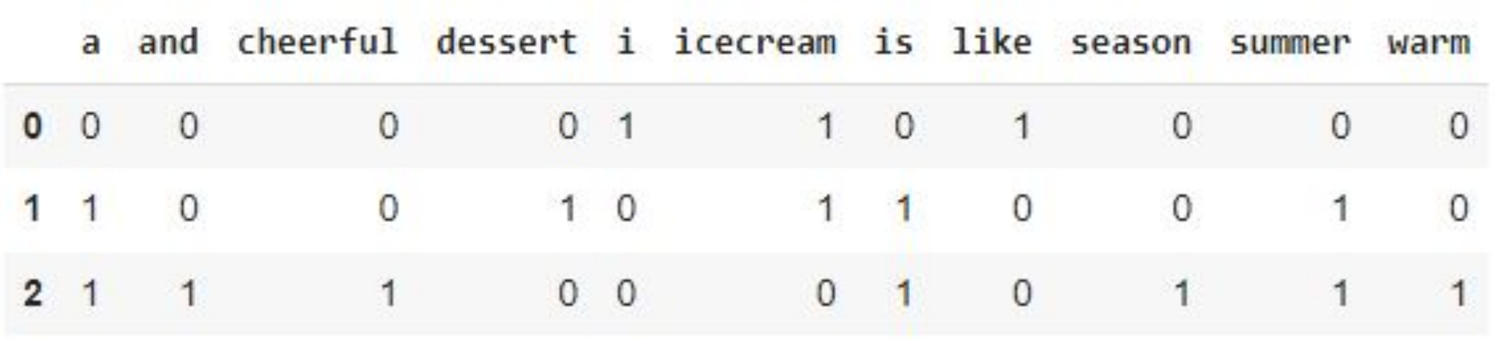
Question Answering architecture



Reading Wikipedia to Answer Open-Domain Questions (2017, machine reading at scale)

Sparse text representation: bag of words

Doc. 0 = “I like icecream”
Doc. 1 = “Icecream is a summer dessert”
Doc. 2 = “Summer is a warm and cheerful season”



	a	and	cheerful	dessert	i	icecream	is	like	season	summer	warm
0	0	0	0	0	1	1	0	1	0	0	0
1	1	0	0	1	0	1	1	0	0	1	0
2	1	1	1	0	0	0	1	0	1	1	1

Sparse text representation: TF-IDF

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

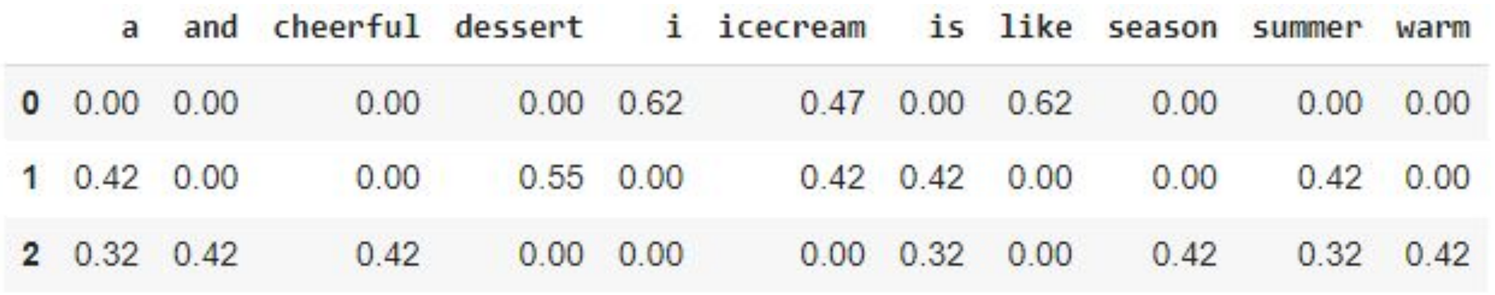
Term x within document y

$tf_{x,y}$ = frequency of x in y

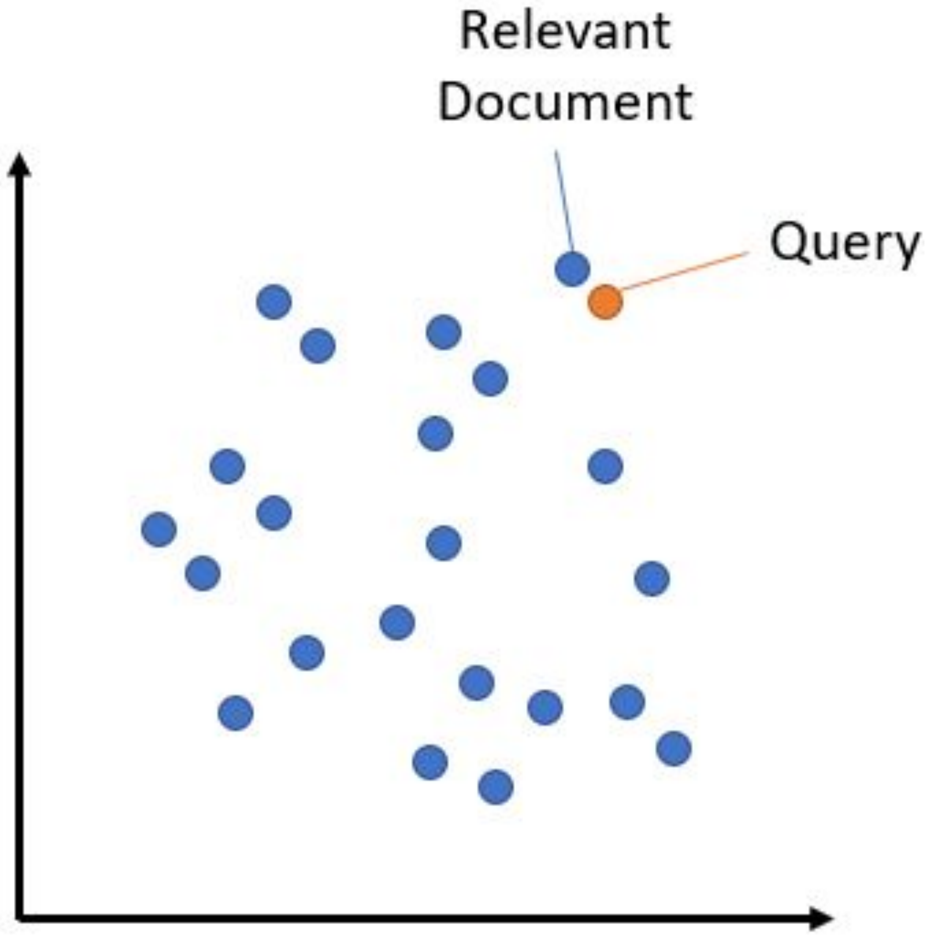
df_x = number of documents containing x

N = total number of documents

Doc. 0 = “I like icecream”
Doc. 1 = “Icecream is a summer dessert”
Doc. 2 = “Summer is a warm and cheerful season”



	a	and	cheerful	dessert	i	icecream	is	like	season	summer	warm
0	0.00	0.00	0.00	0.00	0.62	0.47	0.00	0.62	0.00	0.00	0.00
1	0.42	0.00	0.00	0.55	0.00	0.42	0.42	0.00	0.00	0.42	0.00
2	0.32	0.42	0.42	0.00	0.00	0.00	0.32	0.00	0.42	0.32	0.42



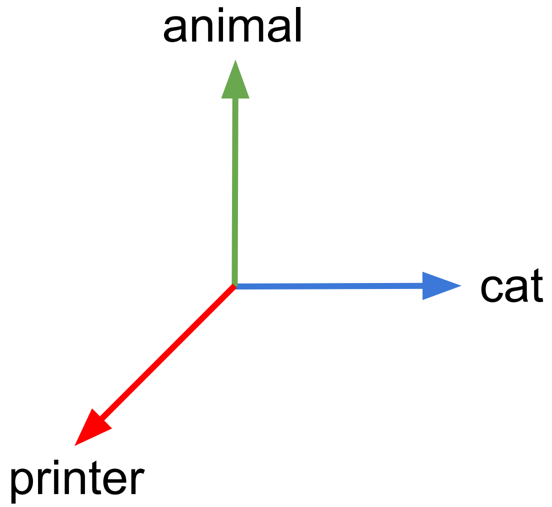
Why do we need vectors?

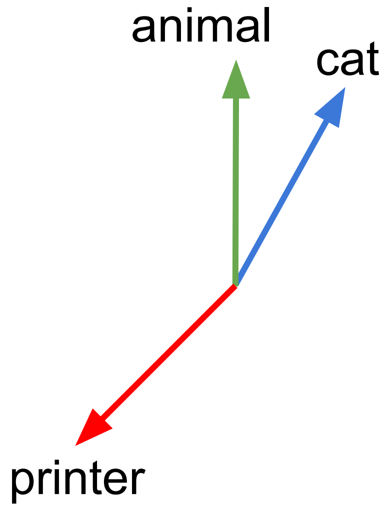
Sparse text representations for retrieval

TF-IDF, BM25

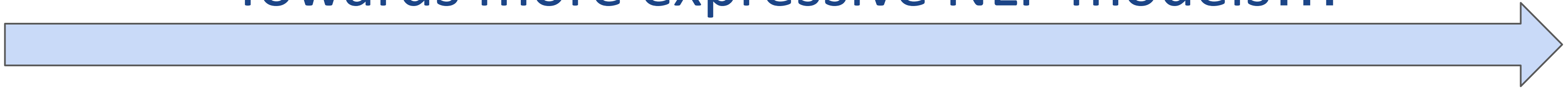
- simple but effective
- don’t need to be trained
- work on any language

- rely on exact keyword matches between query and text (searching for “handbook”, you’ll never find “manual”)





Towards more expressive NLP models…

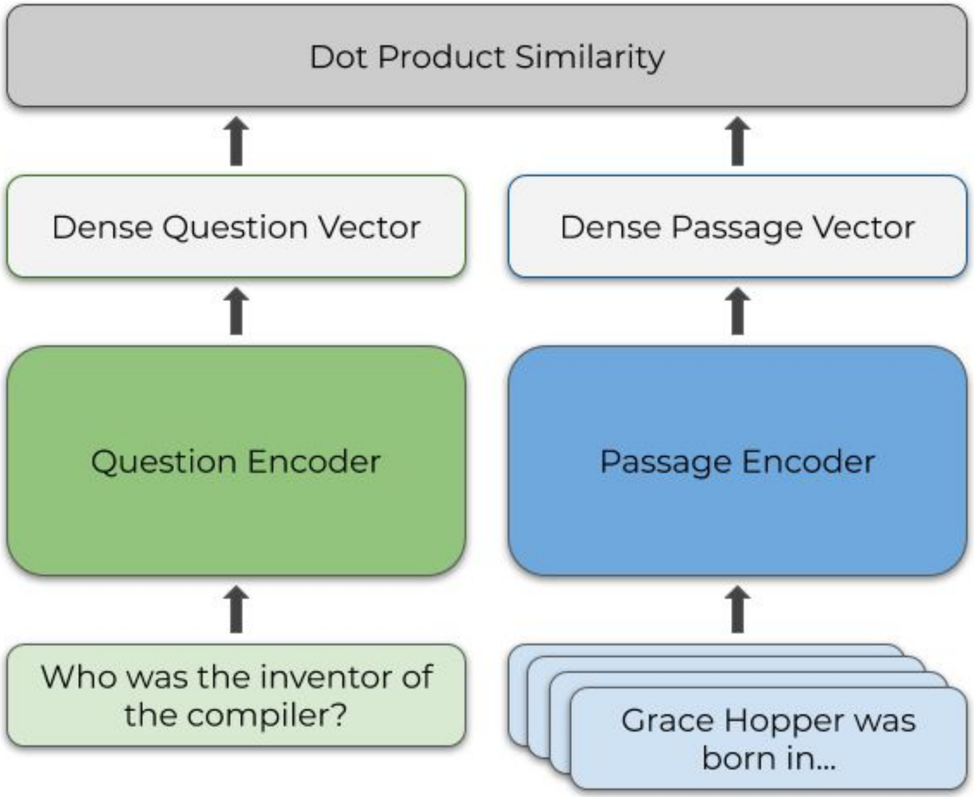


Sparse text representations
(since 1960)

Word embeddings
(Word2vec, 2013)

Transformers
(BERT, 2018)

Dense Passage Retrieval



Doc. 0 = “I like icecream” [0.08517568, 0.74978304, 0.12174767, ..., 0.28093684, 0.78732026, 0.63918763]
Size 768

up to +19% retrieval accuracy, compared to BM25

[Dense Passage Retrieval for Open-Domain Question Answering (2020)]

Another flavor of dense retrieval: Embedding Retrieval

- Only one transformer model to encode documents and queries
- Pretrained models available in Sentence Transformers
- In many cases, Embedding Retrieval works better than Dense Passage Retrieval

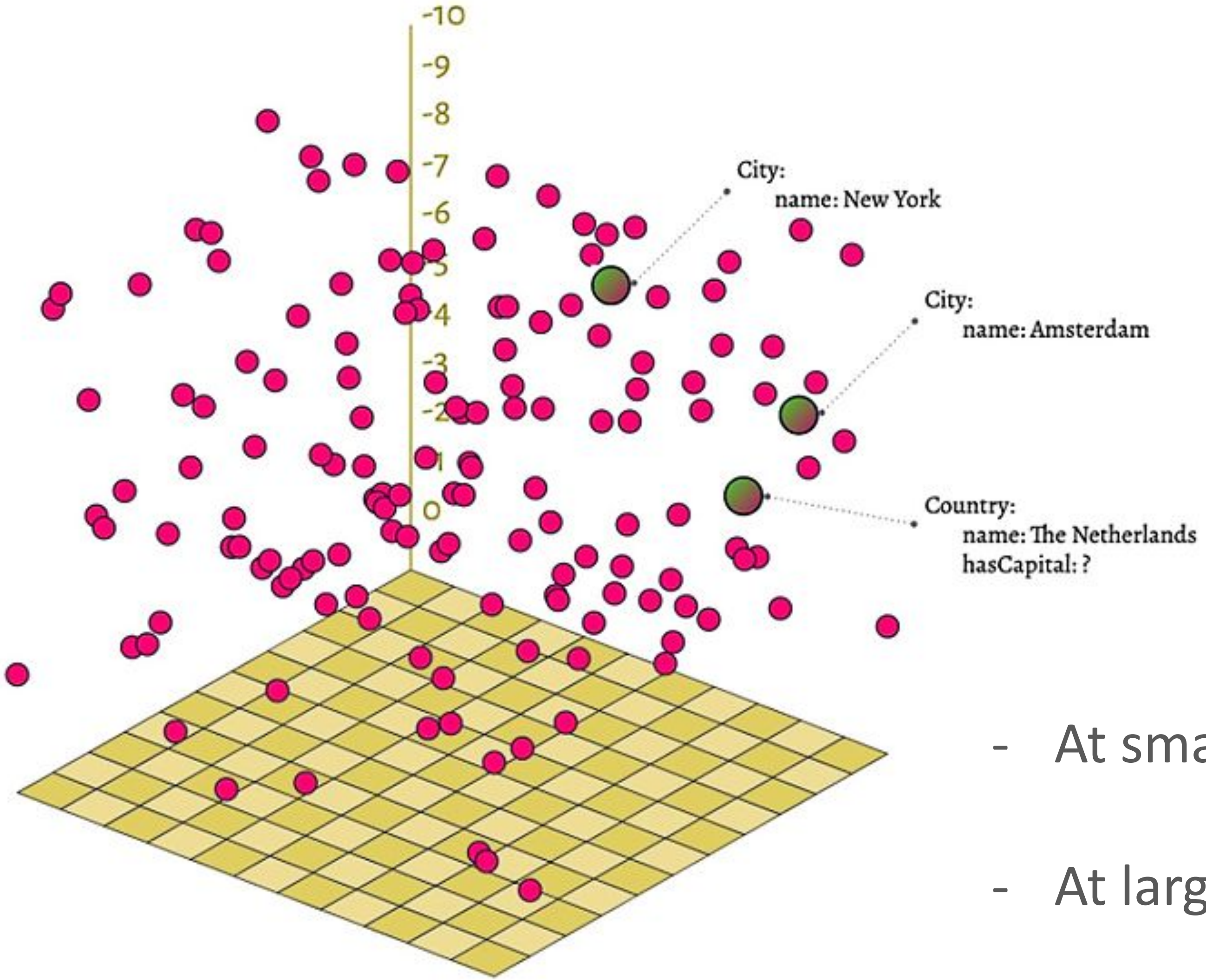
Dense text representations for retrieval

Dense Passage Retrieval, Embedding Retrieval

- capture semantic similarity
- large improvements in retrieval accuracy, compared to sparse representations
- trainable

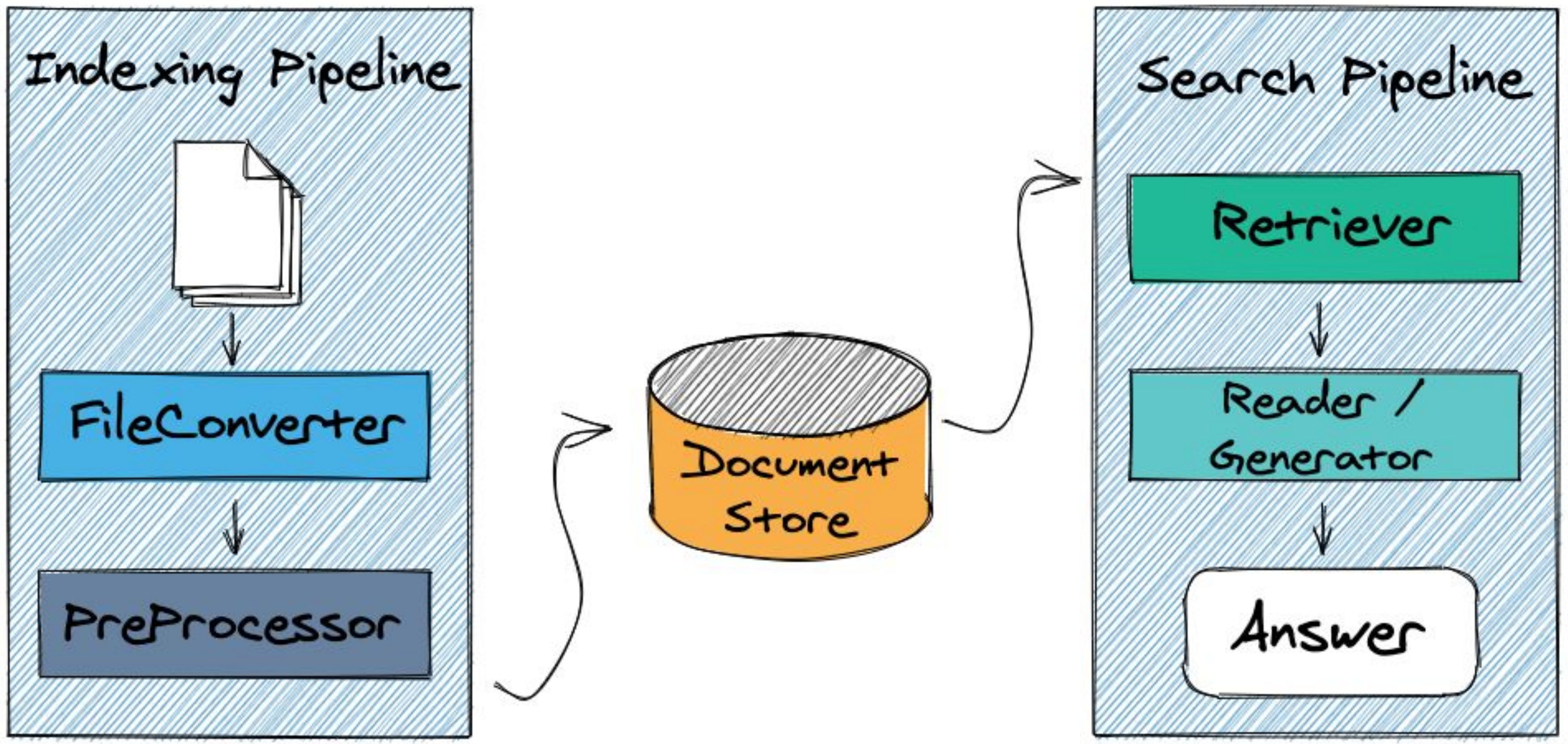
- computationally more heavy

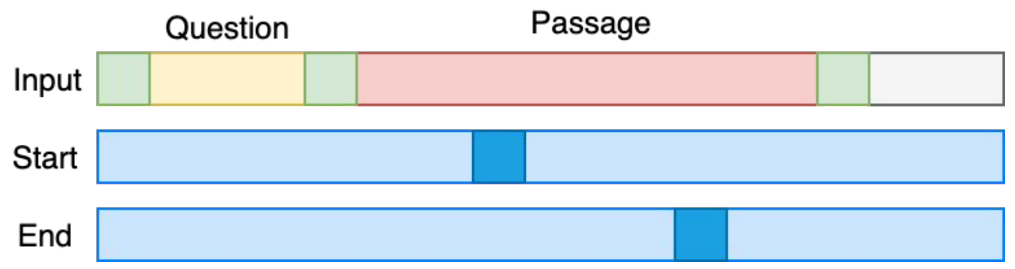
Vector search in a nutshell



- At small scale: KNN search
- At larger scale: ANN search

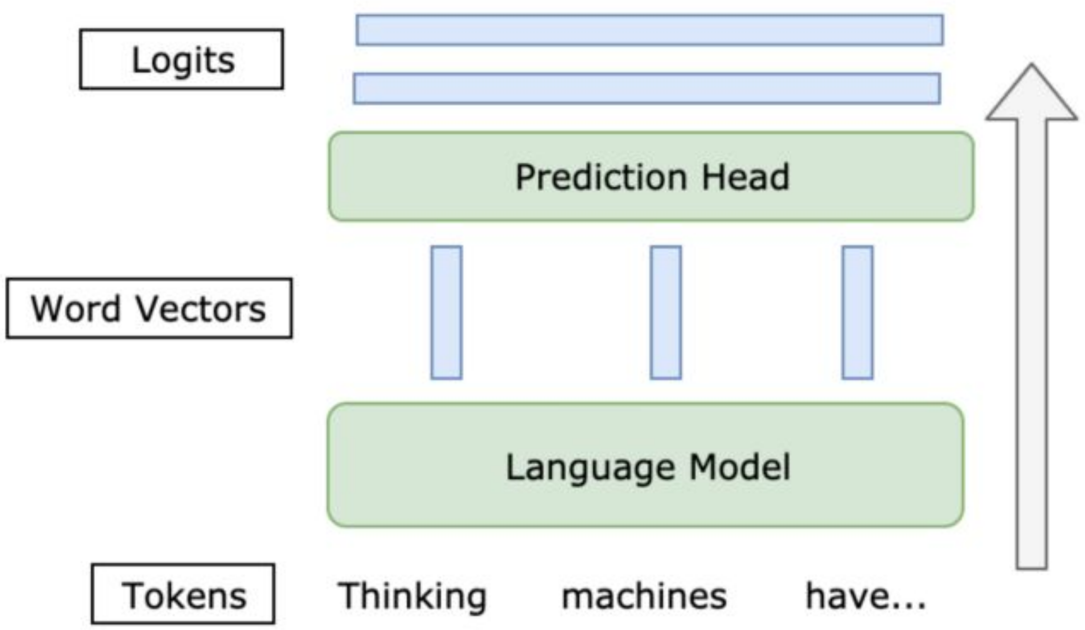
Question Answering architecture





Q: Where is Twin Peaks?

...eriff Harry S. Truman Twin Peaks was a small logging town in **northeastern Washington State**, five miles south of the Canadian border and twelve miles w...



The Reader

Question Answering in Python: Haystack

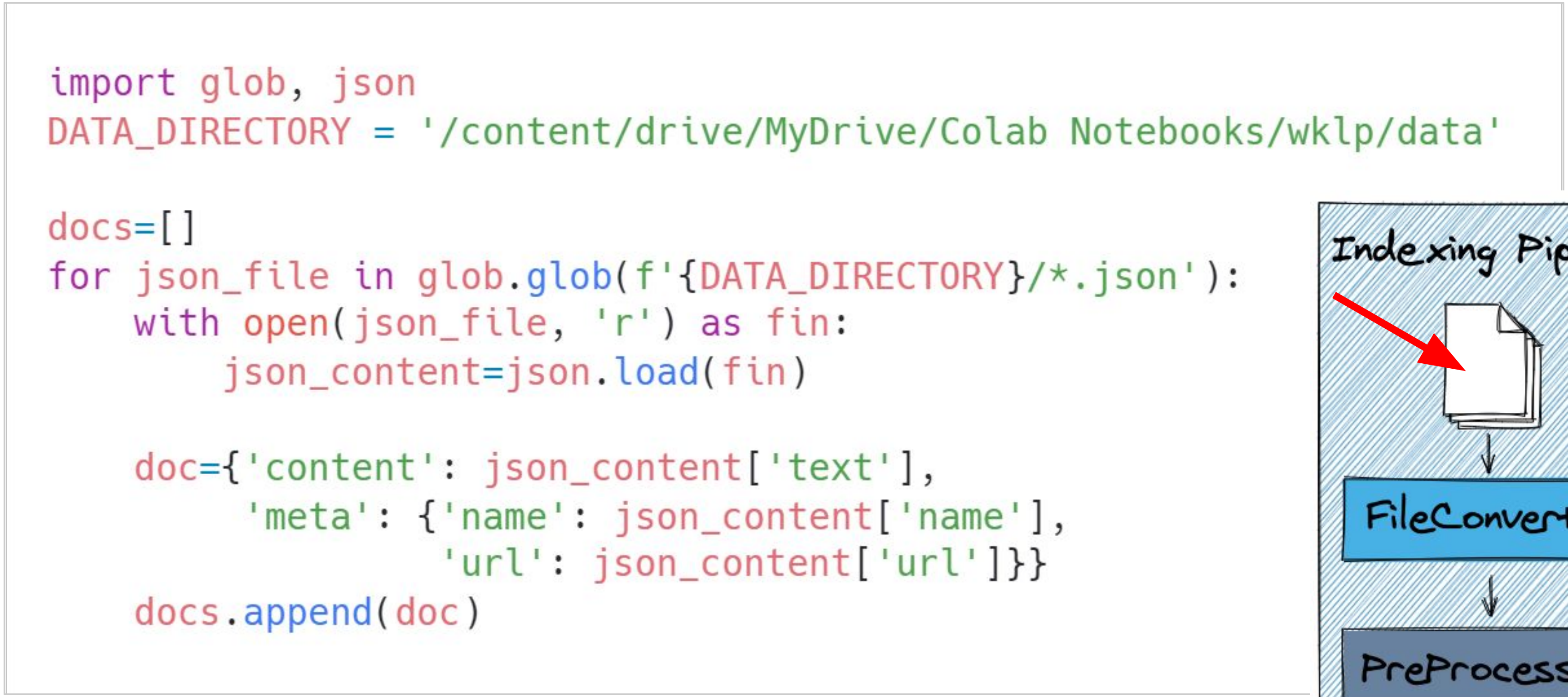
Haystack is an open-source framework for building search systems that work intelligently over large document collections.

Features:

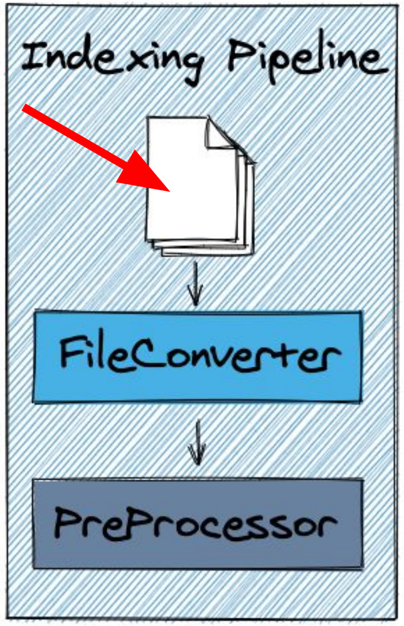
- Modular elements
- Latest models
- Flexible Document Store
- Scalability
- Domain adaptation

haystack by deepset

Building Twin Peaks QA - 1. Load data

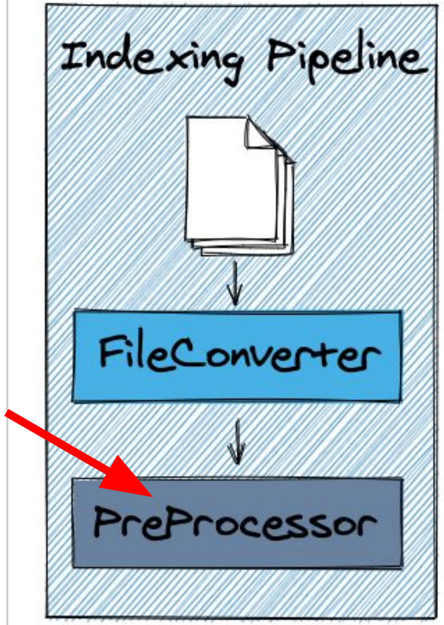


```python
import glob, json
DATA_DIRECTORY = '/content/drive/MyDrive/Colab Notebooks/wklp/data'

docs=[]
for json_file in glob.glob(f'{DATA_DIRECTORY}/*.json'):
    with open(json_file, 'r') as fin:
        json_content=json.load(fin)

    doc={'content': json_content['text'],
         'meta': {'name': json_content['name'],
                  'url': json_content['url']}}
    docs.append(doc)
```



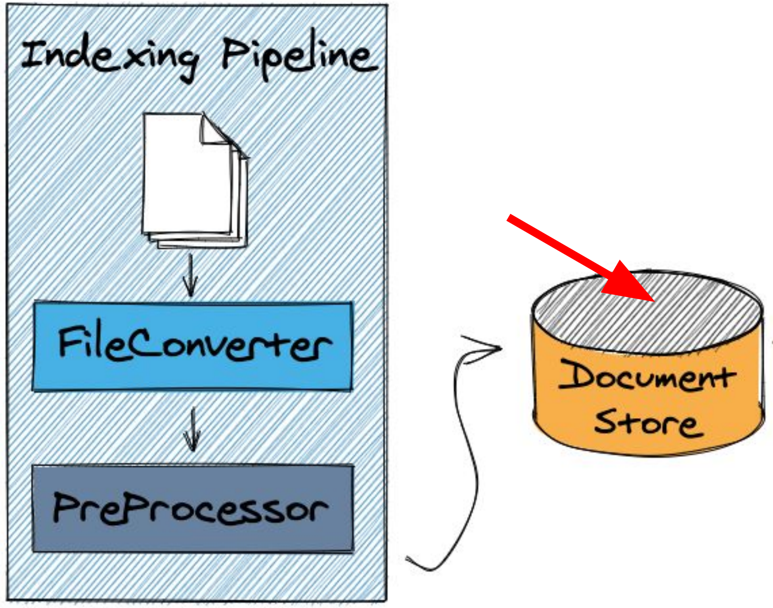
2. Preprocess data

```
# preprocess documents, splitting by chunks of 200 words
from haystack.nodes import PreProcessor
processor = PreProcessor(
    clean_empty_lines=True,
    clean_whitespace=True,
    clean_header_footer=True,
    split_by="word",
    split_length=200,
    split_respect_sentence_boundary=True,
    split_overlap=0,
    language ='en')
preprocessed_docs = processor.process(docs)
```

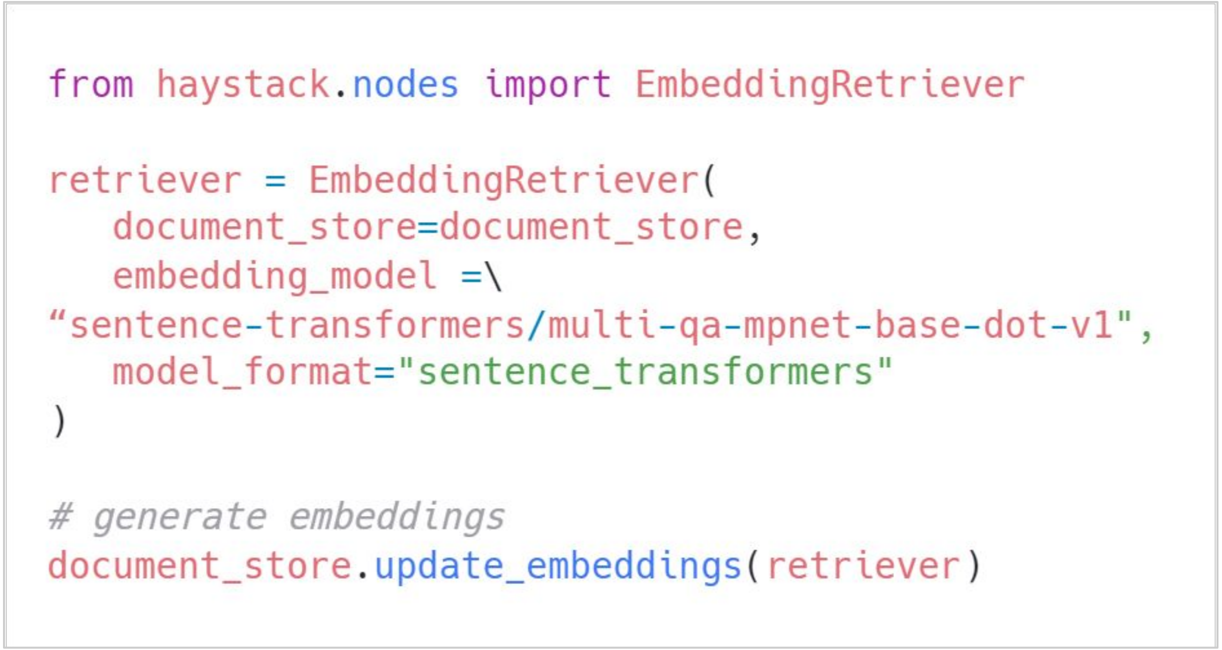


3. Create document store and write documents

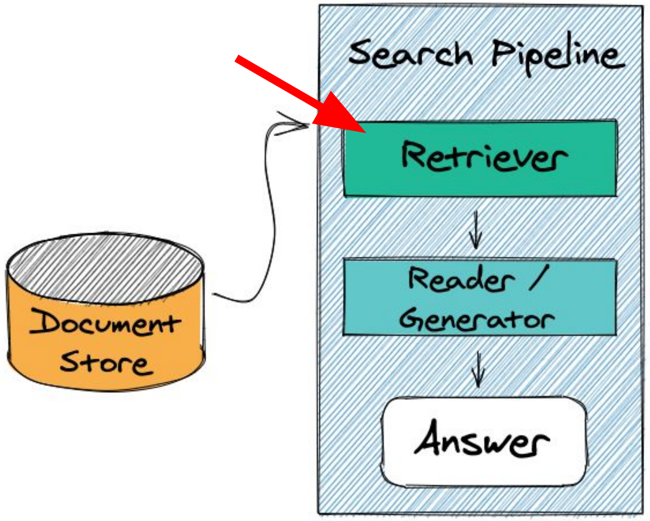
```
from haystack.document_stores
import FAISSDocumentStore

# the document store settings are
# those compatible
# with Embedding Retriever
document_store = \
FAISSDocumentStore(
    similarity="dot_product",
    embedding_dim=768)

# write documents
document_store.write_documents(
  preprocessed_docs)
```



4. Initialize retriever and generate embeddings

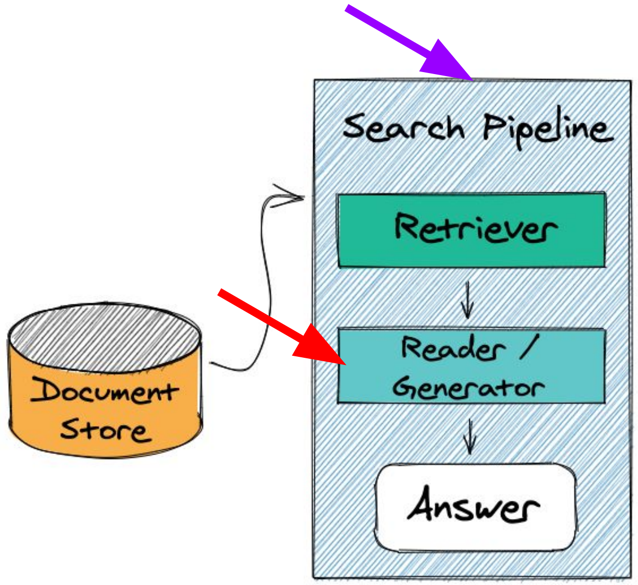


```python
from haystack.nodes import EmbeddingRetriever

retriever = EmbeddingRetriever(
    document_store=document_store,
    embedding_model =\
“sentence-transformers/multi-qa-mpnet-base-dot-v1",
    model_format="sentence_transformers"
)

# generate embeddings
document_store.update_embeddings(retriever)
```

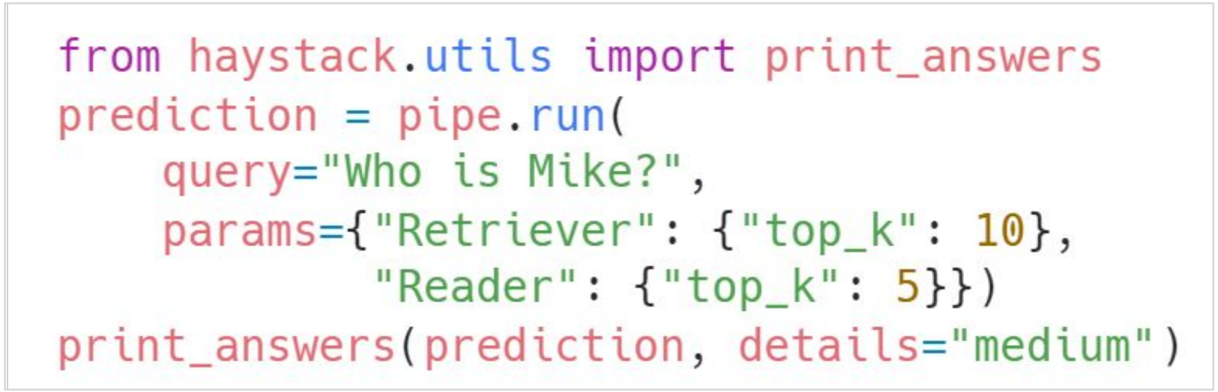


5. Initialize reader and compose QA pipeline

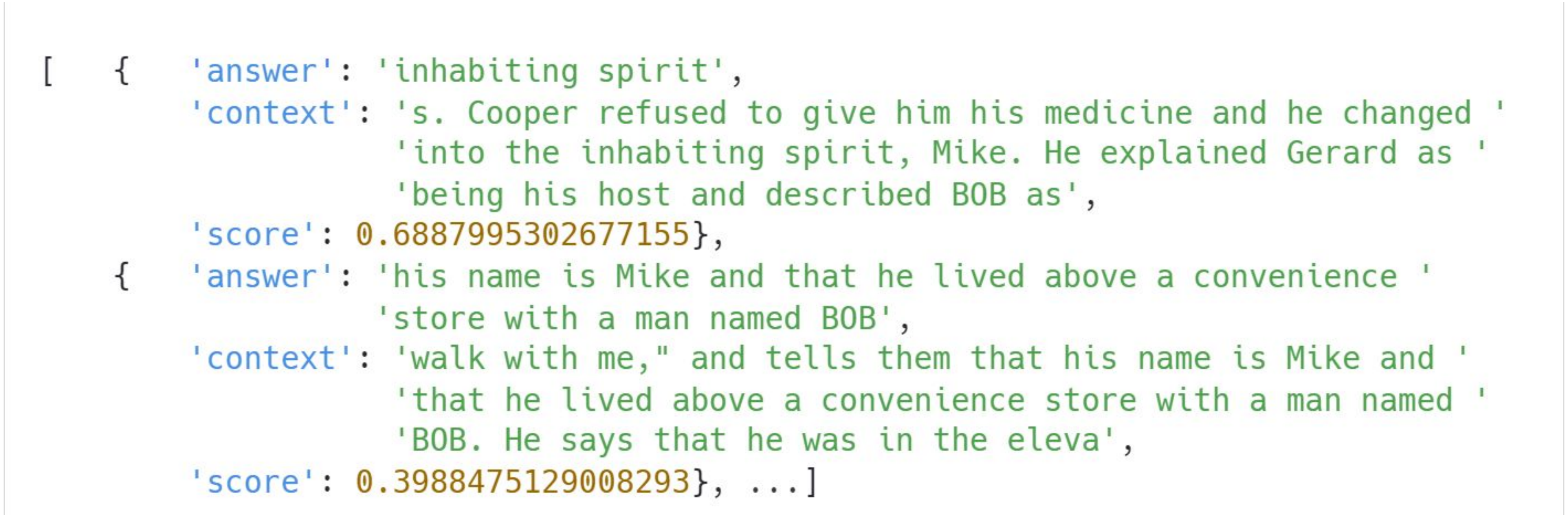
```
from haystack.nodes import FARMReader
reader = FARMReader(model_name_or_path=\
            "deepset/roberta-base-squad2",
            use_gpu=True)

from haystack.pipelines import ExtractiveQAPipeline
pipe = ExtractiveQAPipeline(reader, retriever)
```



6. Let’s try the pipeline!



```python
from haystack.utils import print_answers
prediction = pipe.run(
    query="Who is Mike?",
    params={"Retriever": {"top_k": 10},
            "Reader": {"top_k": 5}})
print_answers(prediction, details="medium")
```

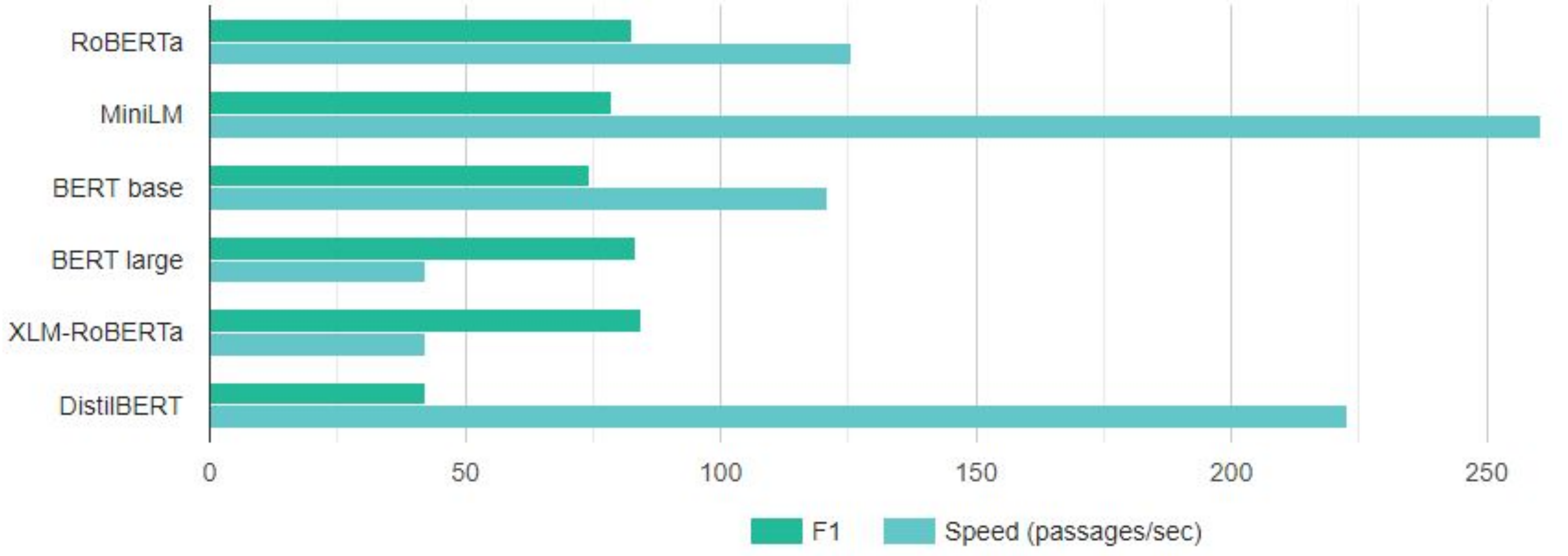


```
[   {   'answer': 'inhabiting spirit',
        'context': 's. Cooper refused to give him his medicine and he changed '
                   'into the inhabiting spirit, Mike. He explained Gerard as '
                   'being his host and described BOB as',
        'score': 0.6887995302677155},
    {   'answer': 'his name is Mike and that he lived above a convenience '
                  'store with a man named BOB',
        'context': 'walk with me," and tells them that his name is Mike and '
                   'that he lived above a convenience store with a man named '
                   'BOB. He says that he was in the eleva',
        'score': 0.3988475129008293}, ...]
```

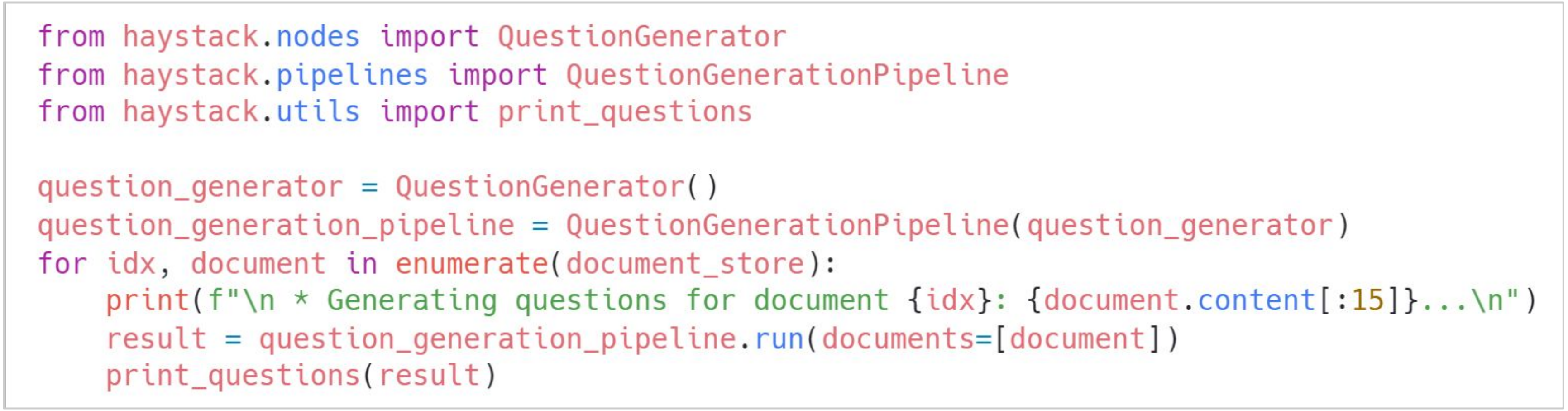
How to choose my components?

Haystack provides extensive documentation, including:

- benchmarks:
  - retriever accuracy/speed
  - reader accuracy/speed
- optimization guide



Question generation



```python
from haystack.nodes import QuestionGenerator
from haystack.pipelines import QuestionGenerationPipeline
from haystack.utils import print_questions

question_generator = QuestionGenerator()
question_generation_pipeline = QuestionGenerationPipeline(question_generator)
for idx, document in enumerate(document_store):
    print(f"\n * Generating questions for document {idx}: {document.content[:15]}...\n")
    result = question_generation_pipeline.run(documents=[document])
    print_questions(result)
```

```
* Generating questions for document 0: Pete Martell...
Generated questions:
 -   Who was the manager of the Packard Sawmill?
 -   What was Pete Martell's job title?
 -   When did Pete marry Catherine Packard?
 -   Pete married what woman in 1958?
 -   What year did Pete become the number one booster for the Twin
Peaks High School football team?
 -   Who was Pete's wife's brother?
```

Haystack: other use cases/features

- Deploy the pipelines as REST APIs
- Chatbot integration
- Query classification
- Generative QA (Retriever-Augmented Generation, LFQA)
- Table QA (TAPAS)
- Summarization
- Translation
- Document classification
- Entity extraction…

haystack

Demo time!

THANKS!

Stefano Fiorucci

Find me on:

- Github (@anakin87)
- Linkedin

hf.co/spaces/anakin87/who-killed-laura-palmer

Spaces

github.com/anakin87/who-killed-laura-palmer

GitHub