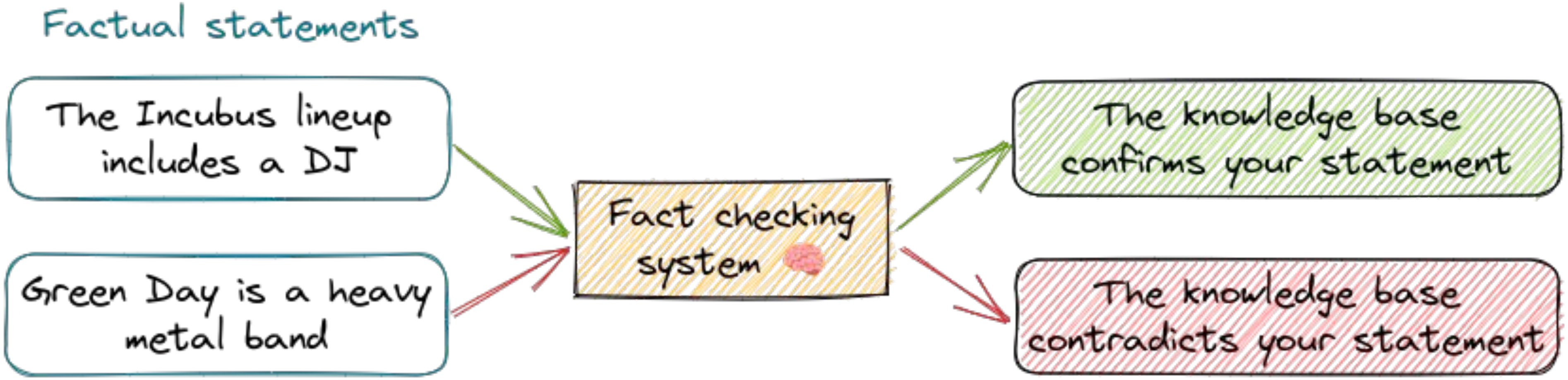
FACT CHECKING ROCKS!

Stefano Fiorucci

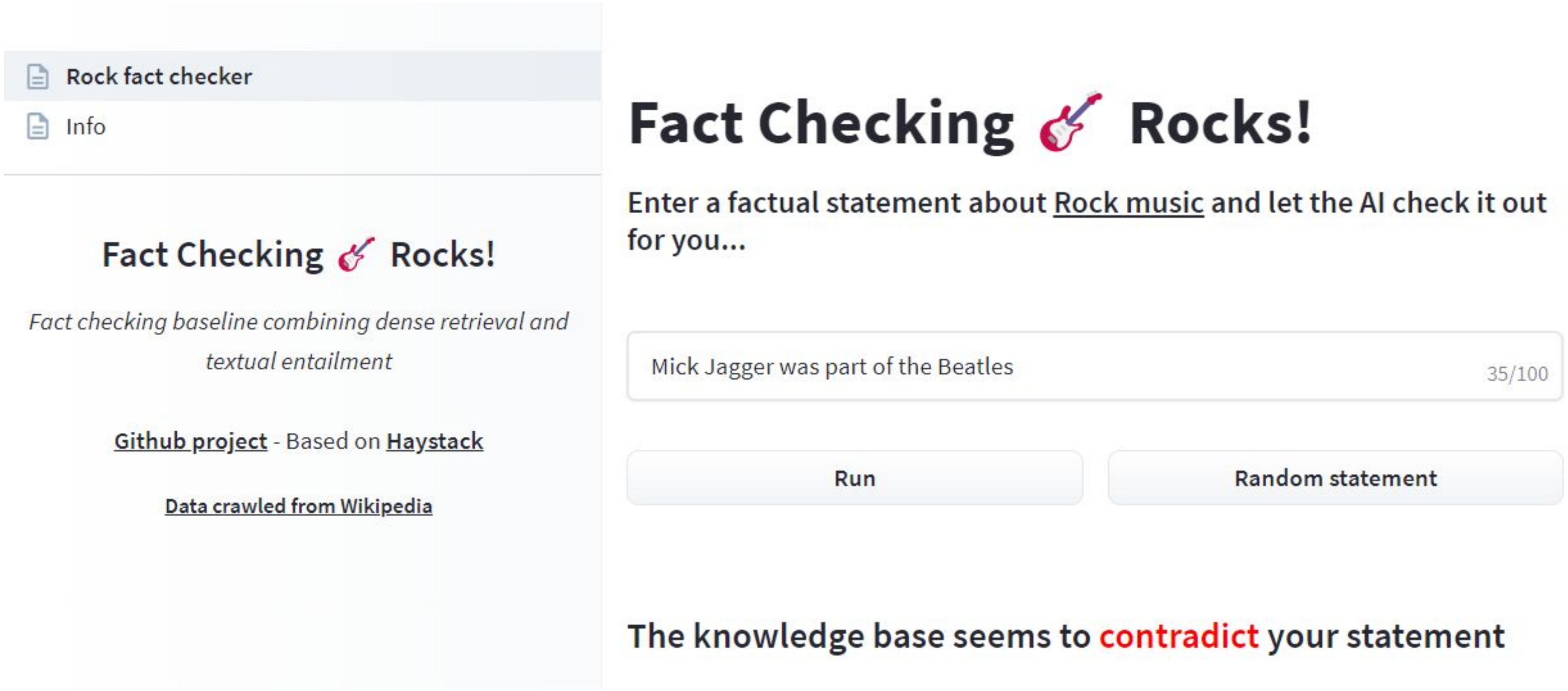
- Machine Learning Engineer (NLP)
- 🔎 Semantic / Neural / Vector Search 🧑‍🚀 explorer

Find me on:
Github/HF (@anakin87) - LinkedIn

Fact Checking 🎸 Rocks!


Factual statements
The Incubus lineup includes a DJ
Green Day is a heavy metal band
Fact checking system
The knowledge base confirms your statement
The knowledge base contradicts your statement

- knowledge from Wikipedia
- modern NLP tools
- zero manual effort


Rock fact checker

Info

Fact Checking 🎸 Rocks!

Fact checking baseline combining dense retrieval and textual entailment

Github project - Based on Haystack

Data crawled from Wikipedia

Fact Checking 🎸 Rocks!

Enter a factual statement about Rock music and let the AI check it out for you...

Mick Jagger was part of the Beatles 35/100

Run | Random statement

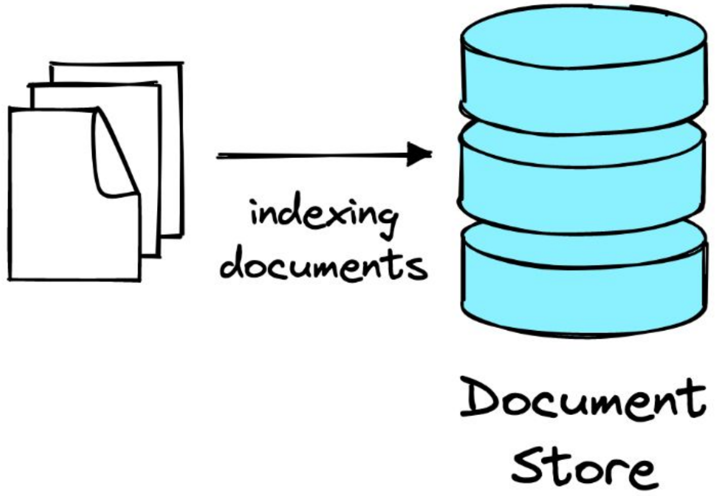
The knowledge base seems to contradict your statement

Agenda

haystack
by deepset

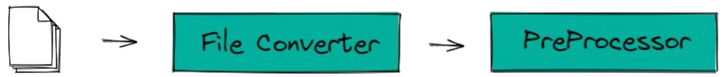
- end-to-end open-source NLP framework
- build
  - search systems
  - (Large) Language Models applications
- modular
- scalable
- customizable

Haystack: Document stores



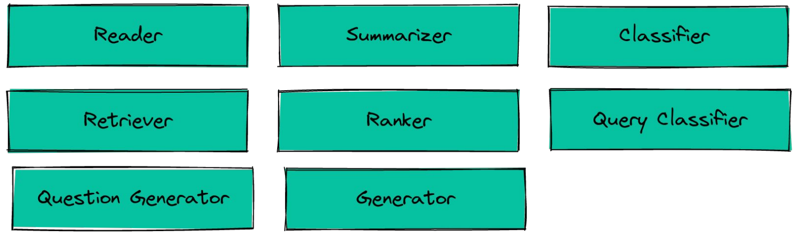
Haystack: Nodes

Data Connectors

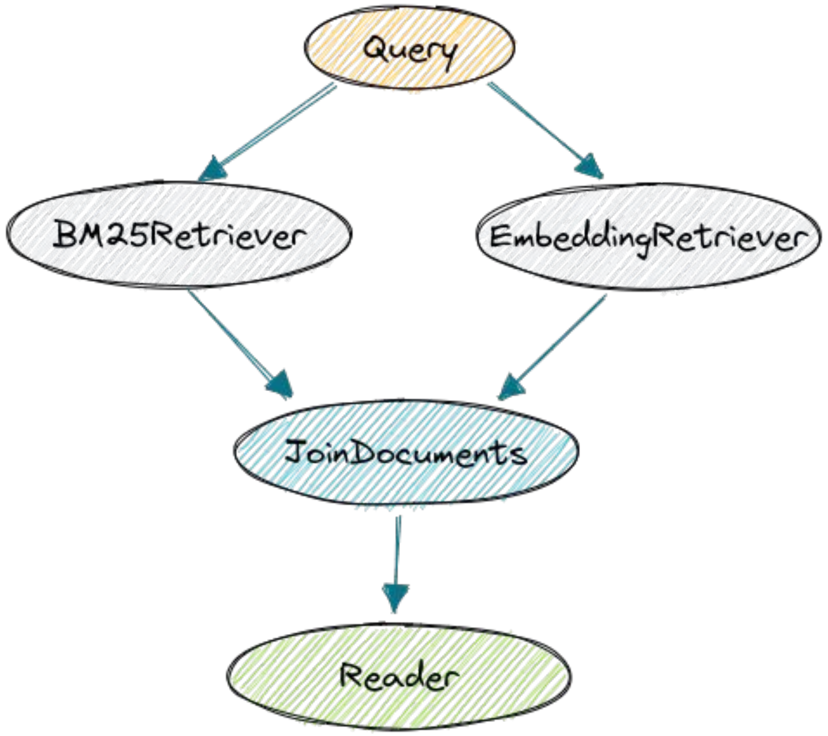


- File converters from txt, pdf, docx, markdown
- Web scraper to turn website into text
- Preprocessor to split long documents, clean text

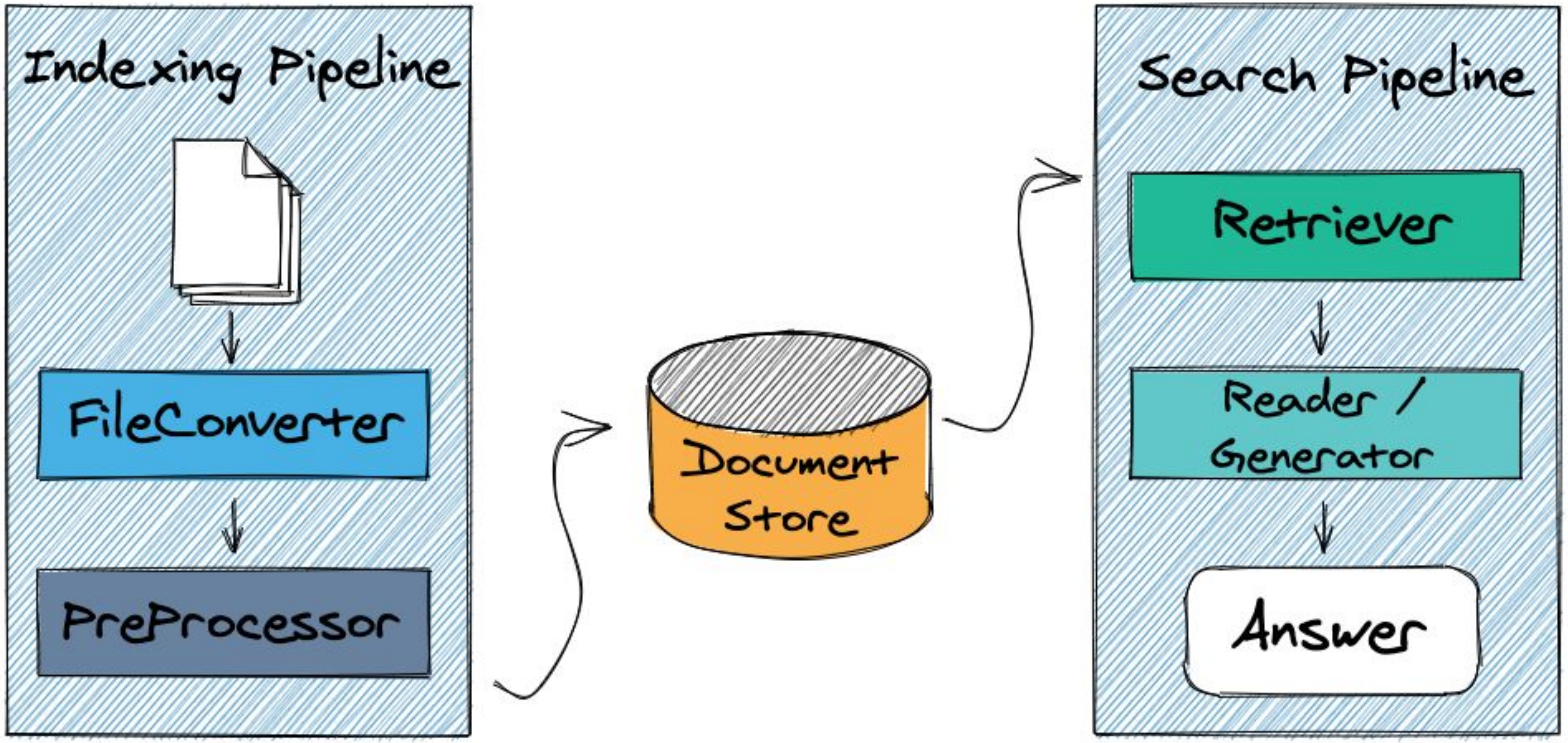
Core NLP Tasks covered



Haystack: Pipelines



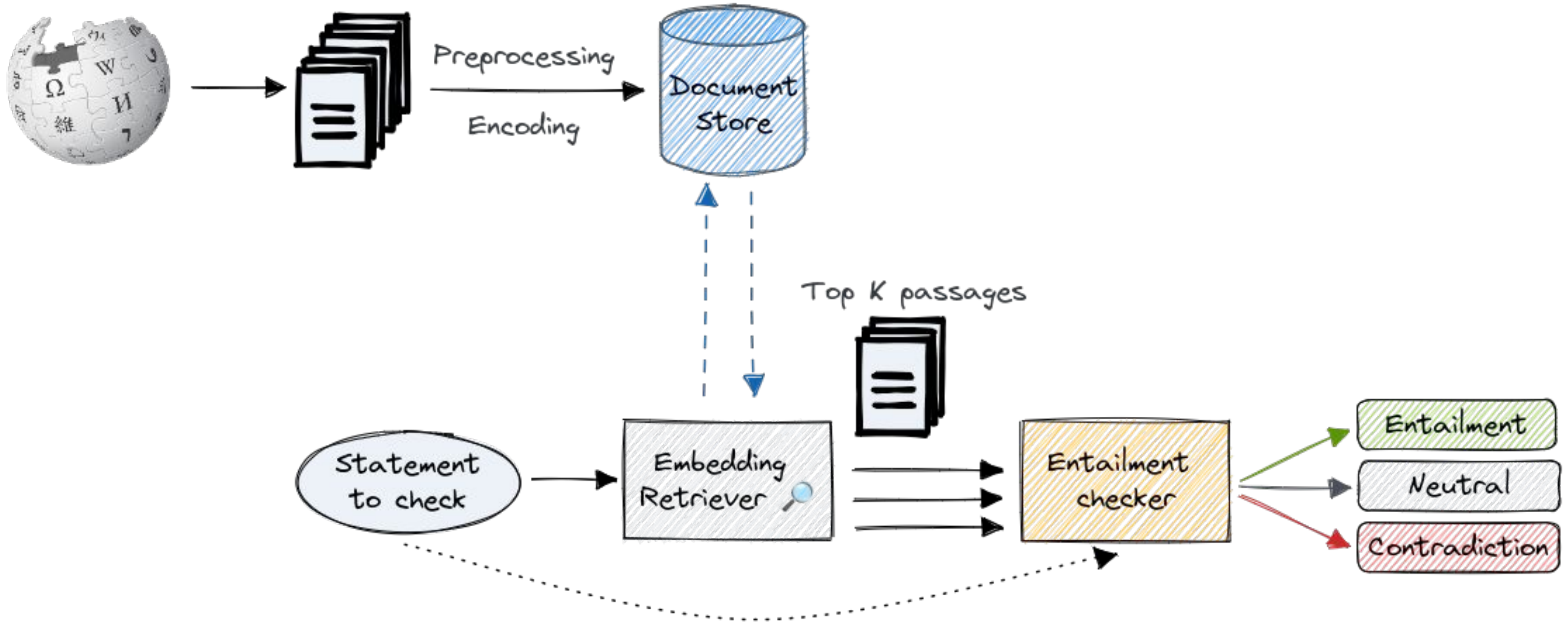
Haystack: Question Answering architecture



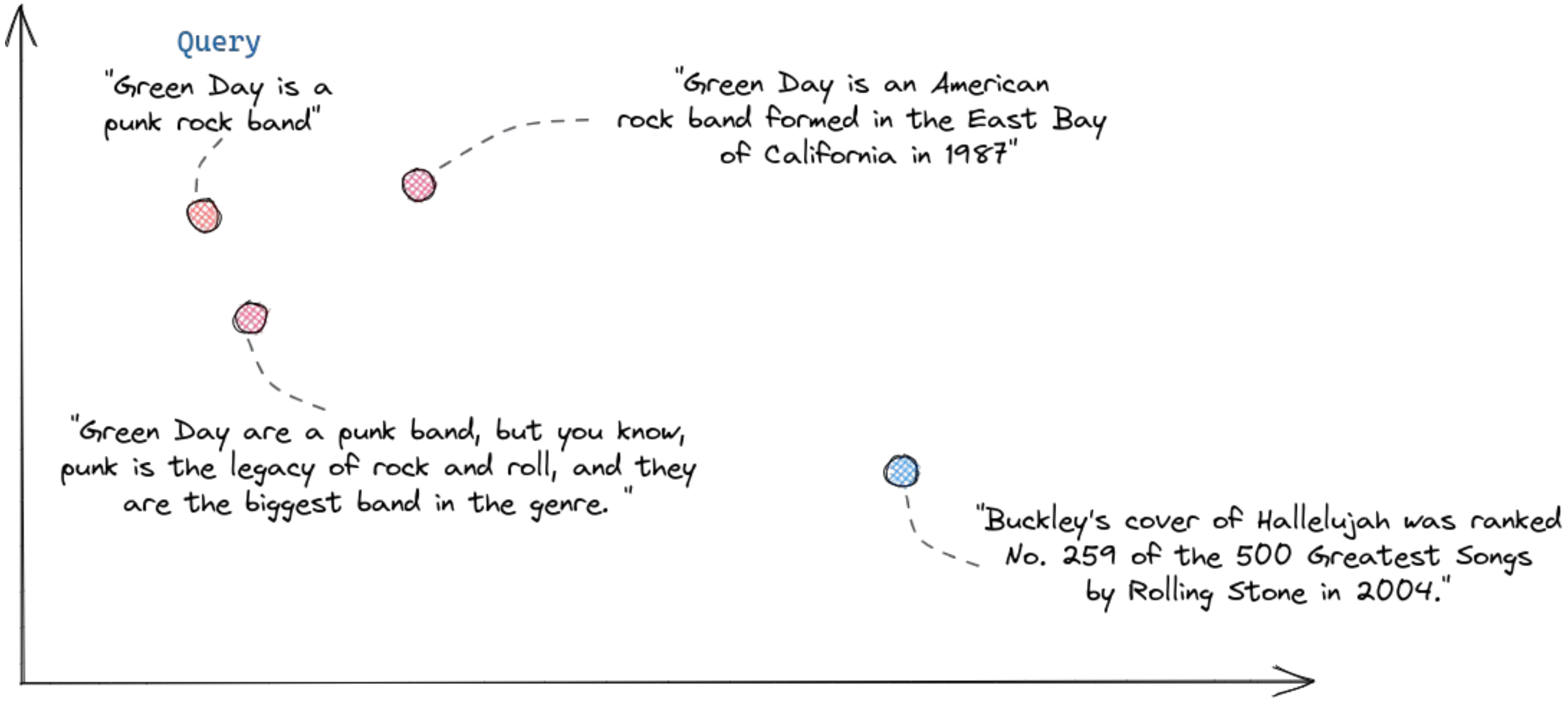
Reading Wikipedia to Answer Open-Domain Questions (2017, machine reading at scale)

Agenda

Fact Checking Rocks: the idea



Embedding Retriever


Query
"Green Day is a punk rock band"
"Green Day is an American rock band formed in the East Bay of California in 1987"
"Green Day are a punk band, but you know, punk is the legacy of rock and roll, and they are the biggest band in the genre."
"Buckley's cover of Hallelujah was ranked No. 259 of the 500 Greatest Songs by Rolling Stone in 2004."

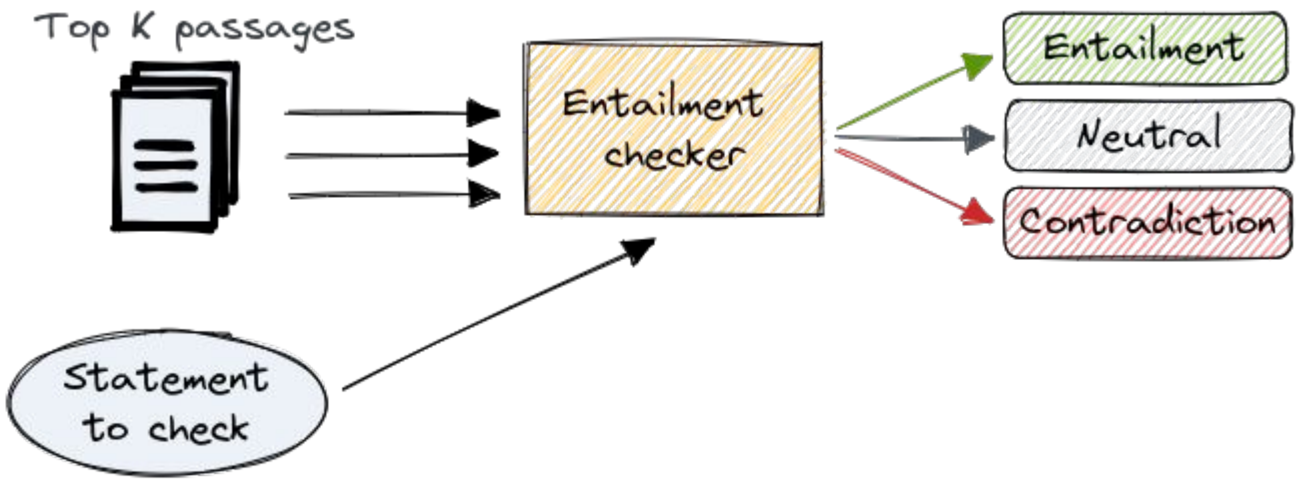
Inspired by Pinecone educational material

Natural Language Inference models

NLI: the task of determining whether a “hypothesis” is true, false, or undetermined given a “premise”.

Premise	Hypothesis	Label
A man inspects the uniform of a figure in some East Asian country.	The man is sleeping.	contradiction
An older and younger man smiling.	Two men are smiling and laughing at the cats playing on the floor.	neutral
A soccer game with multiple males playing.	Some men are playing a sport.	entailment

Entailment Checker node



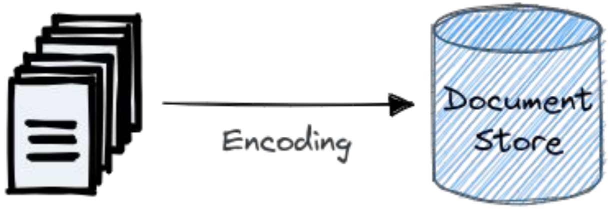
- compute the textual entailment for every passage
- aggregate the textual entailment scores
- empirical consideration: apply a threshold

Agenda

1. Load and preprocess data

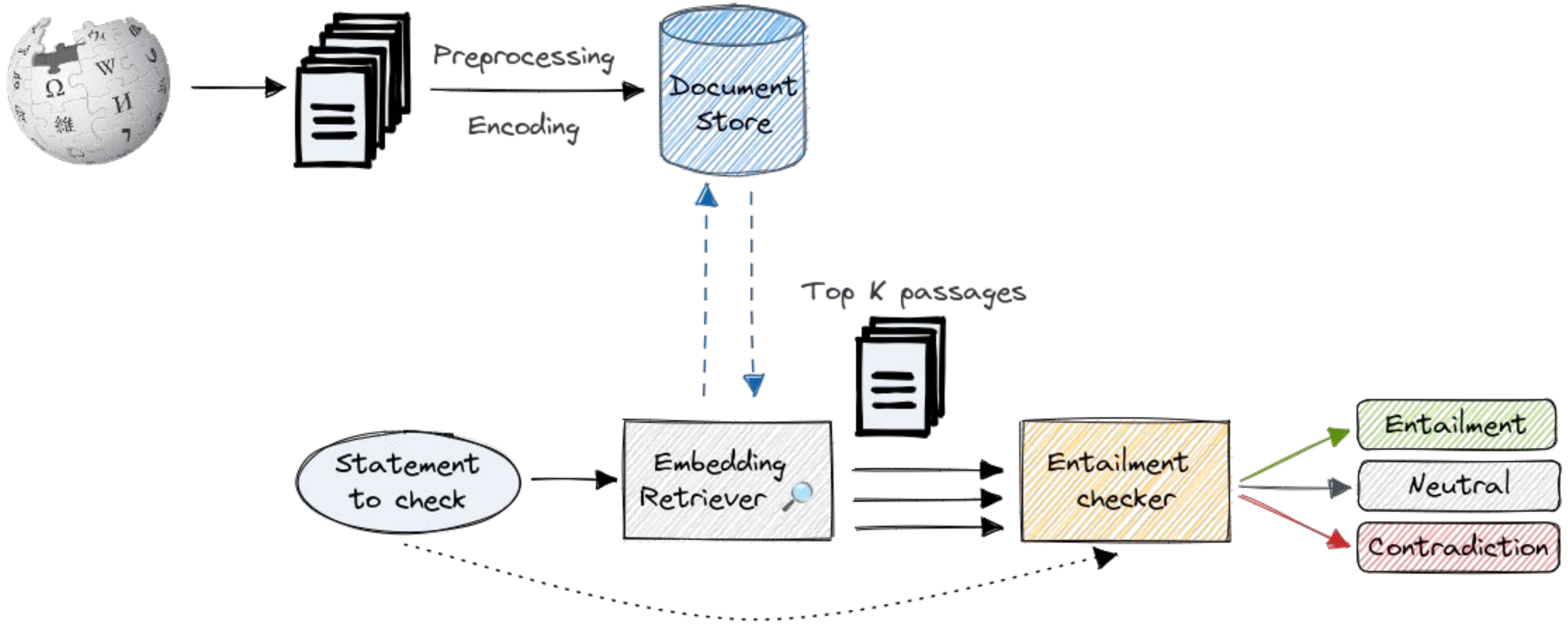
```python
import glob, json
from haystack.nodes import PreProcessor

docs = []
for json_file in glob.glob("rock_wiki/*.json"):
    with open(json_file, "r") as fin:
        doc = json.load(fin)
    docs.append(doc)

# preprocess documents, splitting by chunks of 2 sentences
processor = PreProcessor(
    clean_empty_lines=True,
    clean_whitespace=True,
    clean_header_footer=True,
    split_by="sentence",
    split_length=2,
    split_respect_sentence_boundary=False,
    split_overlap=0,
    language="en",
)
preprocessed_docs = processor.process(docs)
# select only documents with at least 10 words.
# Otherwise, the documents are not very informative
preprocessed_docs = [doc for doc in preprocessed_docs
                     if len(doc.content.split()) >= 10]
```

2. Encode and write documents



```python
from haystack.document_stores import FAISSDocumentStore
from haystack.nodes import EmbeddingRetriever

# 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)

retriever = EmbeddingRetriever(
    document_store=document_store,
    embedding_model="sentence-transformers/msmarco-distilbert-base-tas-b",
    model_format="sentence_transformers",
    embed_meta_fields=["name"],
)
# generate embeddings
document_store.update_embeddings(retriever)
```

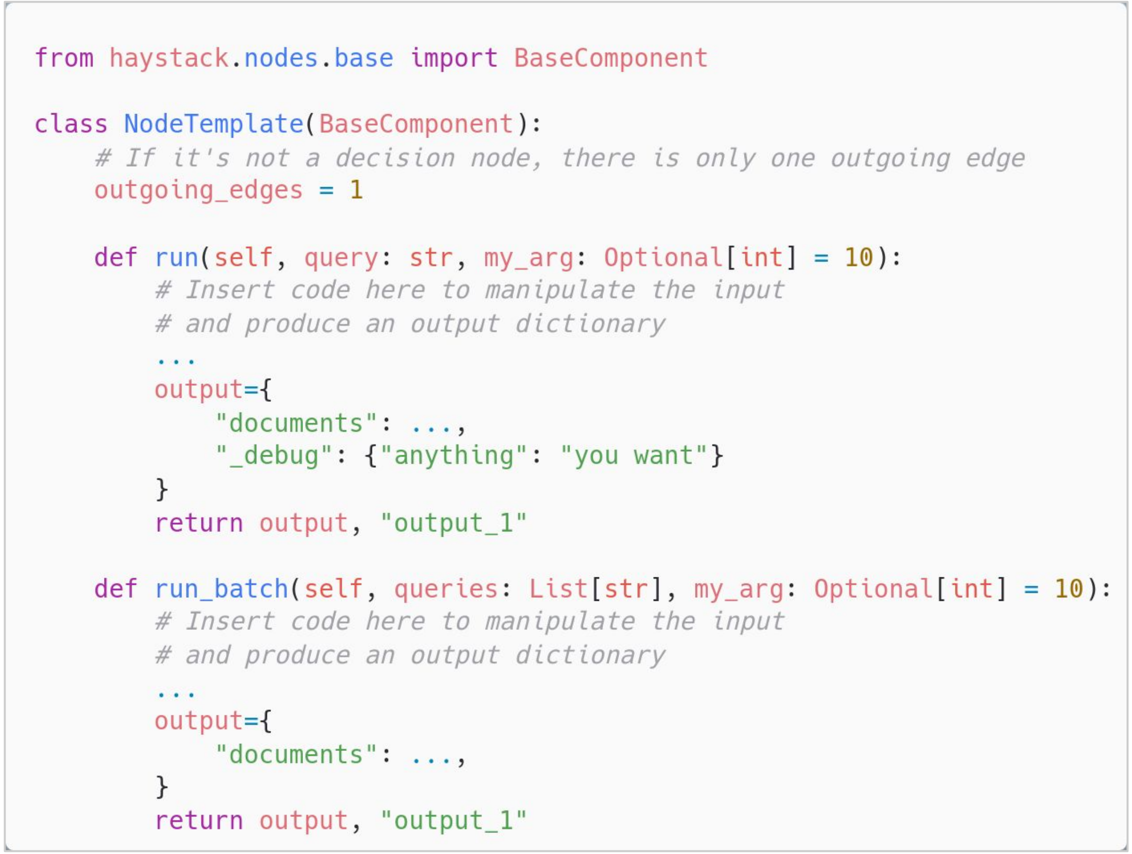
Fact Checking Rocks: the idea



Adding custom nodes to Haystack

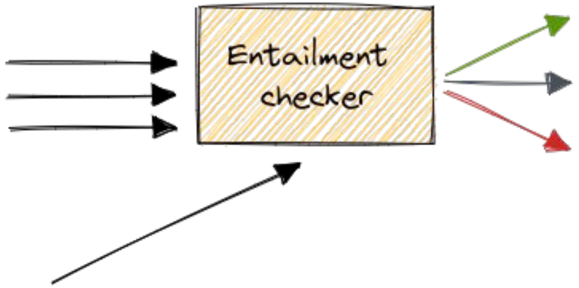
From Haystack docs:

1. Create a new class that inherits from BaseComponent
2. Set outgoing_edges as a class attribute
3. Define a run() method, which returns a tuple
4. Define a run_batch() method, which returns a tuple



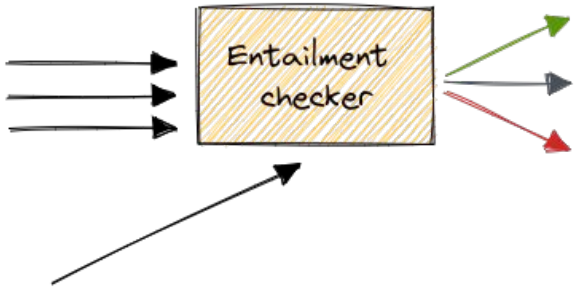
```python
from haystack.nodes.base import BaseComponent

class NodeTemplate(BaseComponent):
    # If it's not a decision node, there is only one outgoing edge
    outgoing_edges = 1

    def run(self, query: str, my_arg: Optional[int] = 10):
        # Insert code here to manipulate the input
        # and produce an output dictionary
        ...
        output={
            "documents": ...,
            "_debug": {"anything": "you want"}
        }
        return output, "output_1"

    def run_batch(self, queries: List[str], my_arg: Optional[int] = 10):
        # Insert code here to manipulate the input
        # and produce an output dictionary
        ...
        output={
            "documents": ...,
        }
        return output, "output_1"
```

3. Entailment Checker node: *__init__*



```python
from typing import List, Optional
from transformers import AutoModelForSequenceClassification, AutoTokenizer, AutoConfig
import torch
from haystack.nodes.base import BaseComponent
from haystack.modeling.utils import initialize_device_settings
from haystack.schema import Document

class EntailmentChecker(BaseComponent):
    outgoing_edges = 1

    def __init__(
        self,
        model_name_or_path: str = "roberta-large-mnli",
        model_version: Optional[str] = None,
        tokenizer: Optional[str] = None,
        use_gpu: bool = True,
        batch_size: int = 16,
        entailment_contradiction_threshold: float = 0.5,
    ):
        super().__init__()
        self.devices, _ = initialize_device_settings(use_cuda=use_gpu, multi_gpu=False)

        tokenizer = tokenizer or model_name_or_path
        self.tokenizer = AutoTokenizer.from_pretrained(tokenizer)
        self.model = AutoModelForSequenceClassification.from_pretrained(
            pretrained_model_name_or_path=model_name_or_path, revision=model_version
        )
        self.batch_size = batch_size
        self.entailment_contradiction_threshold = entailment_contradiction_threshold
        self.model.to(str(self.devices[0]))

        id2label = AutoConfig.from_pretrained(model_name_or_path).id2label
        self.labels = [id2label[k].lower() for k in sorted(id2label)]
        if "entailment" not in self.labels:
            raise ValueError(
                "The model config must contain entailment value in the id2label dict."
            )
```

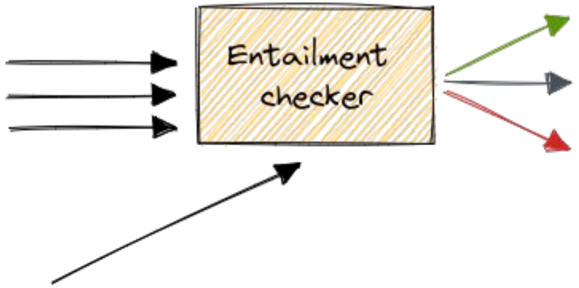
3. Entailment Checker node: *get_entailment*



```python
def get_entailment(self, premise, hypotesis):
    with torch.inference_mode():
        inputs = self.tokenizer(
            f"{premise}{self.tokenizer.sep_token}{hypotesis}",
          return_tensors="pt"
        ).to(self.devices[0])

        out = self.model(**inputs)

        logits = out.logits

        probs = (
            torch.nn.functional.softmax(logits, dim=-1)[0, :]
          .detach().cpu().numpy()
        )
    entailment_dict = {k.lower(): v for k, v in zip(self.labels, probs)}
    return entailment_dict
```

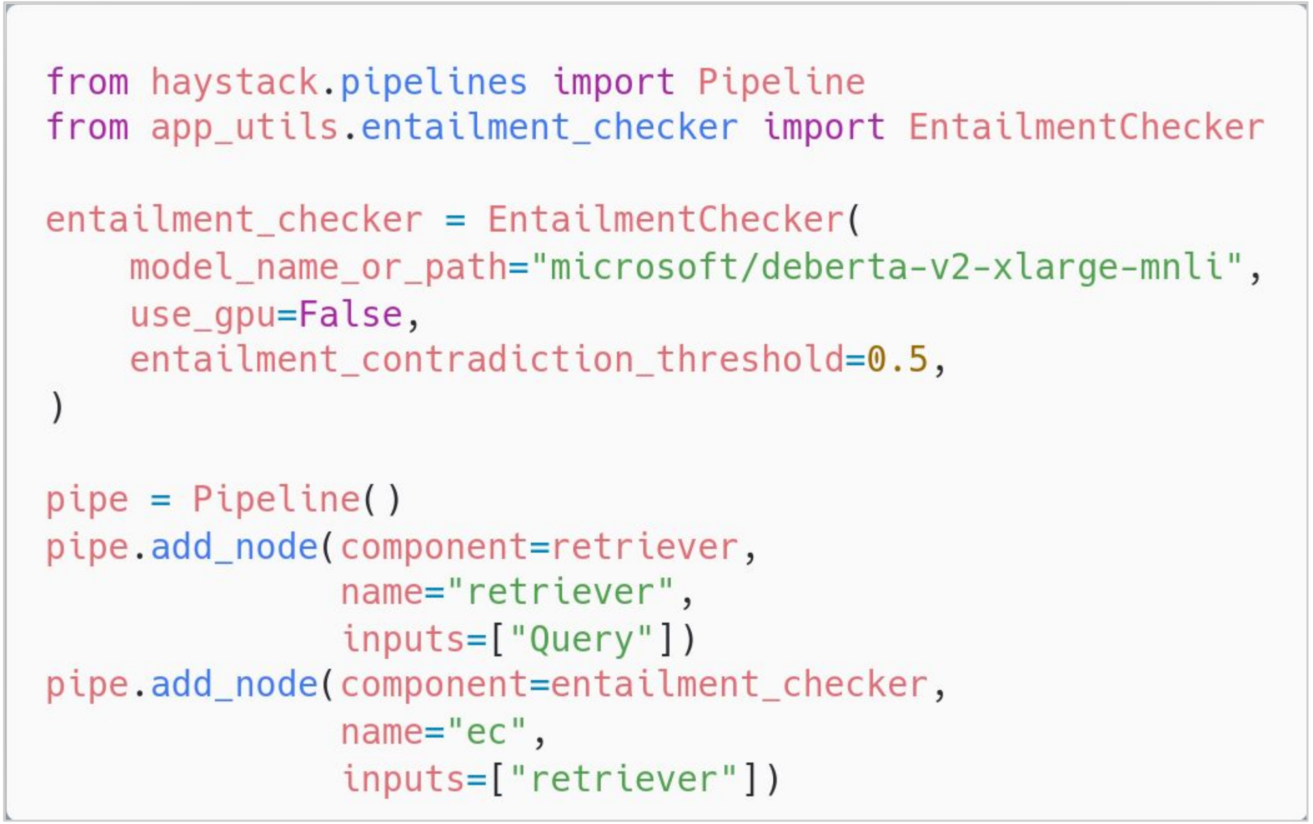
```python
ec = EntailmentChecker()
ec.get_entailment(premise="I have a blue-eyed cat",
                  hypotesis="My cat has yellow eyes")

# {'contradiction': 0.99, 'neutral': 0.01, 'entailment': 0.00}
```

3. Entailment Checker node: *run*



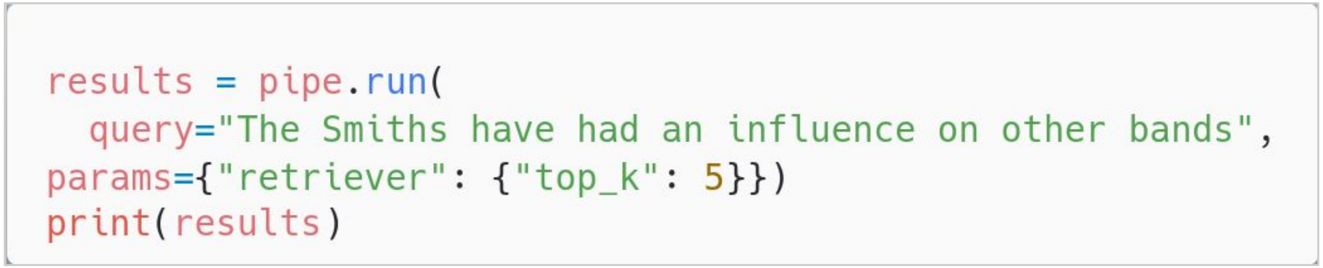
```python
def run(self, query: str, documents: List[Document]):

    scores, agg_con, agg_neu, agg_ent = 0, 0, 0, 0
    for i, doc in enumerate(documents):
        entailment_info = self.get_entailment(premise=doc.content,
                                              hypotesis=query)
        doc.meta["entailment_info"] = entailment_info

        scores += doc.score
        con, neu, ent = (
            entailment_info["contradiction"],
            entailment_info["neutral"],
            entailment_info["entailment"],
        )
        agg_con += con * doc.score
        agg_neu += neu * doc.score
        agg_ent += ent * doc.score

        # if in the first documents there is a strong evidence
        # of entailment/contradiction, there is no need to
        # consider less relevant documents
        if max(agg_con, agg_ent) / scores > self.entailment_contradiction_threshold:
            break

    aggregate_entailment_info = {
        "contradiction": round(agg_con / scores, 2),
        "neutral": round(agg_neu / scores, 2),
        "entailment": round(agg_ent / scores, 2),
    }
    entailment_checker_result = {
        "documents": documents[: i + 1],
        "aggregate_entailment_info": aggregate_entailment_info,
    }
    return entailment_checker_result, "output_1"
```

4. Build the fact-checking pipeline

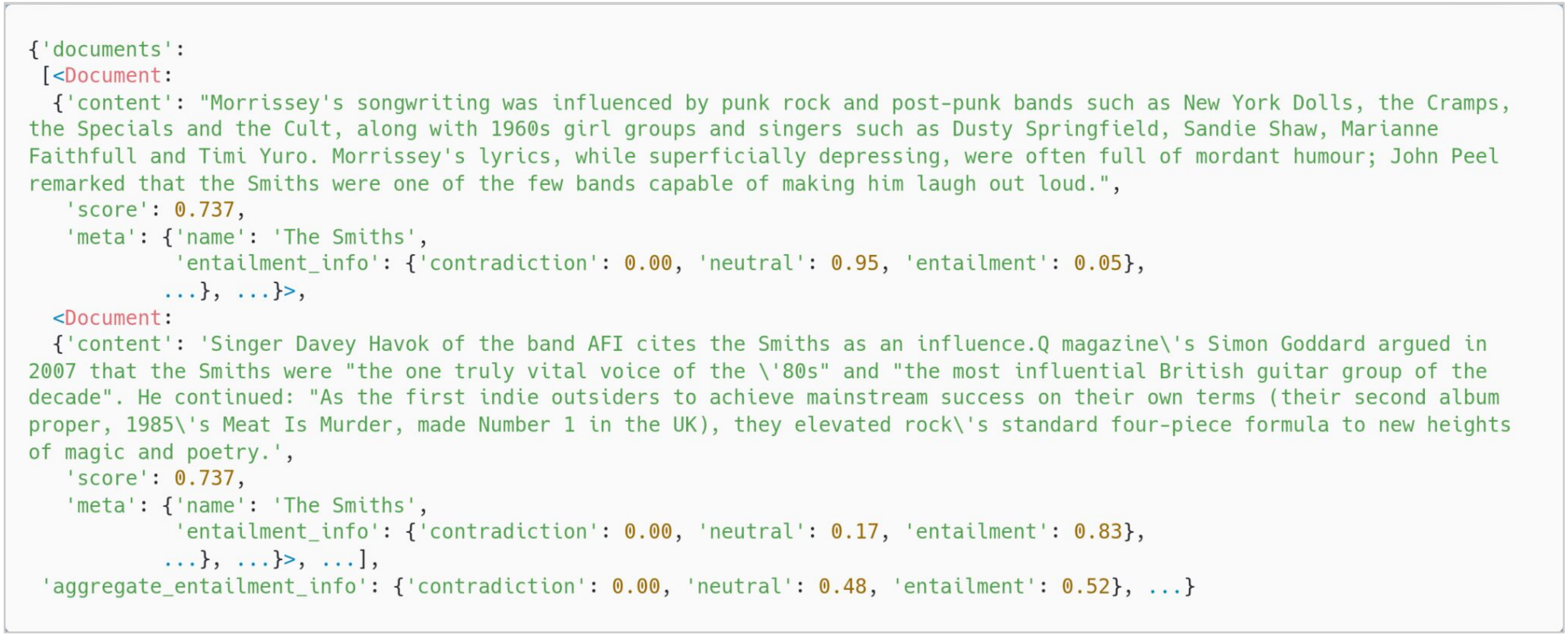


```python
from haystack.pipelines import Pipeline
from app_utils.entailment_checker import EntailmentChecker

entailment_checker = EntailmentChecker(
    model_name_or_path="microsoft/deberta-v2-xlarge-mnli",
    use_gpu=False,
    entailment_contradiction_threshold=0.5,
)

pipe = Pipeline()
pipe.add_node(component=retriever,
              name="retriever",
              inputs=["Query"])
pipe.add_node(component=entailment_checker,
              name="ec",
              inputs=["retriever"])
```

5. Let’s try the pipeline!



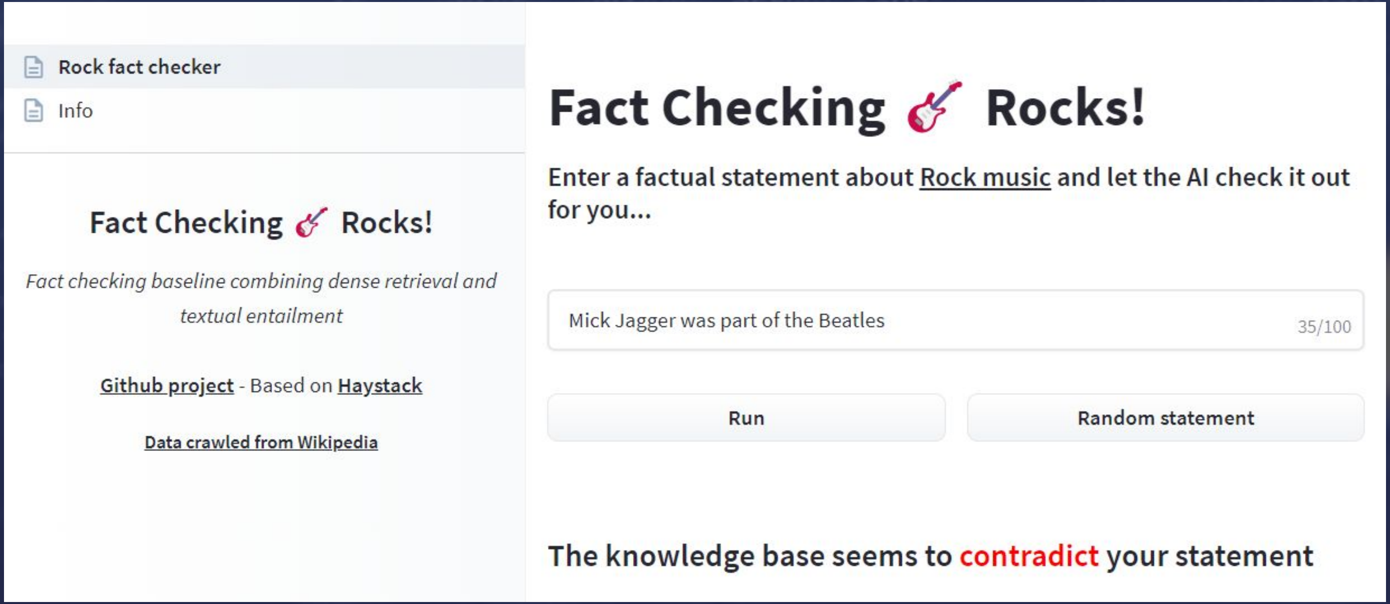
```python
results = pipe.run(
  query="The Smiths have had an influence on other bands",
params={"retriever": {"top_k": 5}})
print(results)
```



```
{'documents':
 [<Document:
  {'content': "Morrissey's songwriting was influenced by punk rock and post-punk bands such as New York Dolls, the Cramps,
the Specials and the Cult, along with 1960s girl groups and singers such as Dusty Springfield, Sandie Shaw, Marianne
Faithfull and Timi Yuro. Morrissey's lyrics, while superficially depressing, were often full of mordant humour; John Peel
remarked that the Smiths were one of the few bands capable of making him laugh out loud.",
    'score': 0.737,
    'meta': {'name': 'The Smiths',
             'entailment_info': {'contradiction': 0.00, 'neutral': 0.95, 'entailment': 0.05},
            ...}, ...}>,
  <Document:
  {'content': 'Singer Davey Havok of the band AFI cites the Smiths as an influence.Q magazine\'s Simon Goddard argued in
2007 that the Smiths were "the one truly vital voice of the \'80s" and "the most influential British guitar group of the
decade". He continued: "As the first indie outsiders to achieve mainstream success on their own terms (their second album
proper, 1985\'s Meat Is Murder, made Number 1 in the UK), they elevated rock\'s standard four-piece formula to new heights
of magic and poetry.',
    'score': 0.737,
    'meta': {'name': 'The Smiths',
             'entailment_info': {'contradiction': 0.00, 'neutral': 0.17, 'entailment': 0.83},
            ...}, ...}>, ...],
 'aggregate_entailment_info': {'contradiction': 0.00, 'neutral': 0.48, 'entailment': 0.52}, ...}
```

Demo time!

hf.co/spaces/anakin87/fact-checking-rocks



Rock fact checker

Info

Fact Checking 🎸 Rocks!

Fact checking baseline combining dense retrieval and textual entailment

Github project - Based on Haystack

Data crawled from Wikipedia

Fact Checking 🎸 Rocks!

Enter a factual statement about Rock music and let the AI check it out for you...

Mick Jagger was part of the Beatles 35/100

Run | Random statement

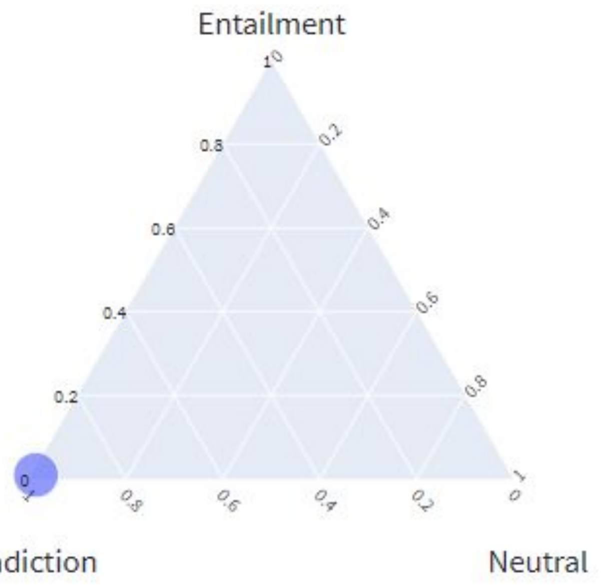
The knowledge base seems to contradict your statement

Elvis Presley is alive

Fact Checking 🎸 Rocks!

The knowledge base seems to contradict your statement

Aggregate entailment information:



```
{
  "contradiction" : 0.99
  "neutral" : 0.01
  "entailment" : 0.01
}
```

Most Relevant snippets:

	Title	Relevance	con	neu	ent	Content
0	Elvis Presley	0.742	0.99	0.01	0.01	Elvis Aaron Presley (January 8, 1935 – August 16, 1977), or simpl

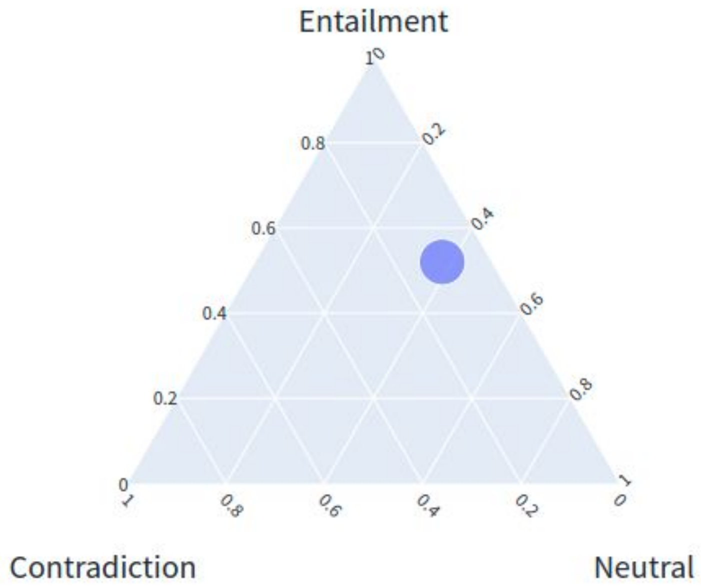
Wikipedia source pages: Elvis Presley

Fact Checking 🎸 Rocks!

The Beach Boys were involved with the Manson family

The knowledge base seems to confirm your statement

Aggregate entailment information:



```
{
  "contradiction" : 0.1
  "neutral" : 0.38
  "entailment" : 0.52
}
```

Most Relevant snippets:

	Title	Relevance	con	neu	ent	Content
0	The Beach Boys	0.749	0.01	0.61	0.39	According to Jon Parks, the band's tour man
1	The Beach Boys	0.747	0.42	0.58	0.00	According to Leaf, "The entire Wilson family
2	The Beach Boys	0.740	0.00	0.08	0.92	Drawing from the Beach Boys' associations
3	The Beach Boys	0.737	0.06	0.41	0.53	Dennis then proposed that Manson be signe
4	The Beach Boys	0.736	0.01	0.23	0.77	In June 1968, Dennis befriended Charles Ma

Agenda

Fact Checking Rocks: limitations ⚠️

- no statement detection
- Wikipedia is taken as a source of truth
- no guarantee that the best text passages for NLI emerge from semantic similarity
- no organic evaluation was performed

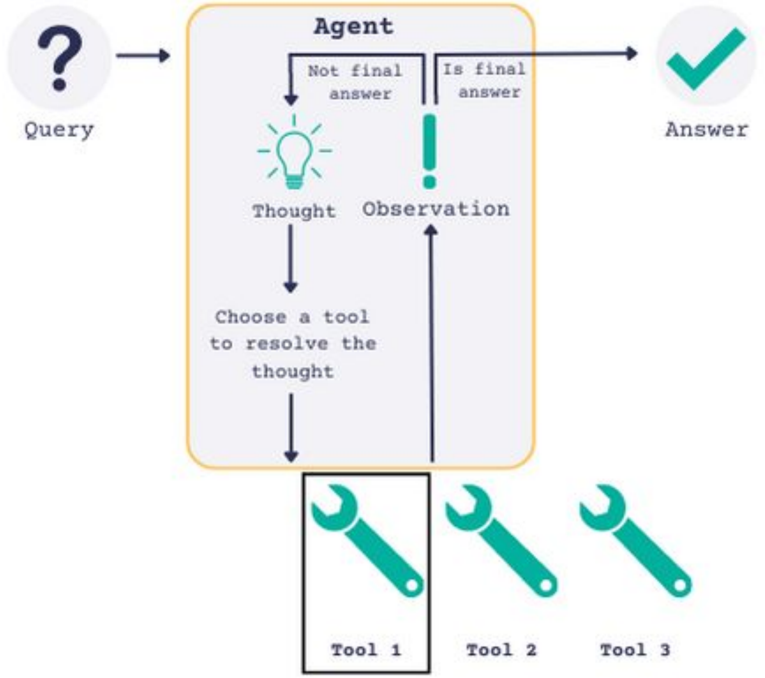
Fact Checking Rocks: how to improve ✨

- expand the knowledge base 📚📚📚
- adapt the retriever to the domain, using [Generative Pseudo Labelling]

Building 🏗️ NLP applications

- (Large) Language Models 🧠 have strong text comprehension/generation abilities
- Their knowledge is generic and not easily updated over time
- Combine them with 🔎 Retrieval systems!

LLMs in Haystack: PromptNode and Agents🕵️



https://github.com/anakin87/try-agents-haystack

https://docs.haystack.deepset.ai/docs/agent

haystack
by deepset

https://haystack.deepset.ai

https://discord.gg/haystack

hf.co/spaces/anakin87/fact-checking-rocks

github.com/anakin87/fact-checking-rocks

Stefano Fiorucci
- Github/HF (@anakin87)
- LinkedIn