RAG Application using Langchain

October 31, 2024

[4]:
```
!pip install weaviate-client
```

```
Collecting weaviate-client
  Downloading weaviate_client-4.9.0-py3-none-any.whl.metadata (3.6 kB)
Requirement already satisfied: requests<3.0.0,>=2.30.0 in
/usr/local/lib/python3.10/dist-packages (from weaviate-client) (2.32.3)
Collecting httpx<=0.27.0,>=0.25.0 (from weaviate-client)
  Downloading httpx-0.27.0-py3-none-any.whl.metadata (7.2 kB)
Collecting validators==0.34.0 (from weaviate-client)
  Downloading validators-0.34.0-py3-none-any.whl.metadata (3.8 kB)
Collecting authlib<1.3.2,>=1.2.1 (from weaviate-client)
  Downloading Authlib-1.3.1-py2.py3-none-any.whl.metadata (3.8 kB)
Requirement already satisfied: pydantic<3.0.0,>=2.5.0 in
/usr/local/lib/python3.10/dist-packages (from weaviate-client) (2.9.2)
Requirement already satisfied: grpcio<2.0.0,>=1.57.0 in
/usr/local/lib/python3.10/dist-packages (from weaviate-client) (1.64.1)
Collecting grpcio-tools<2.0.0,>=1.57.0 (from weaviate-client)
  Downloading grpcio_tools-1.67.1-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (5.3 kB)
Collecting grpcio-health-checking<2.0.0,>=1.57.0 (from weaviate-client)
  Downloading grpcio_health_checking-1.67.1-py3-none-any.whl.metadata (1.1 kB)
Requirement already satisfied: cryptography in /usr/local/lib/python3.10/dist-
packages (from authlib<1.3.2,>=1.2.1->weaviate-client) (43.0.3)
Collecting protobuf<6.0dev,>=5.26.1 (from grpcio-health-
checking<2.0.0,>=1.57.0->weaviate-client)
  Downloading protobuf-5.28.3-cp38-abi3-manylinux2014_x86_64.whl.metadata (592
bytes)
Collecting grpcio<2.0.0,>=1.57.0 (from weaviate-client)
  Downloading grpcio-1.67.1-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (3.9 kB)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-
packages (from grpcio-tools<2.0.0,>=1.57.0->weaviate-client) (75.1.0)
Requirement already satisfied: anyio in /usr/local/lib/python3.10/dist-packages
(from httpx<=0.27.0,>=0.25.0->weaviate-client) (3.7.1)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-
packages (from httpx<=0.27.0,>=0.25.0->weaviate-client) (2024.8.30)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.10/dist-
packages (from httpx<=0.27.0,>=0.25.0->weaviate-client) (1.0.6)
```

```
Requirement already satisfied: idna in /usr/local/lib/python3.10/dist-packages
(from httpx<=0.27.0,>=0.25.0->weaviate-client) (3.10)
Requirement already satisfied: sniffio in /usr/local/lib/python3.10/dist-
packages (from httpx<=0.27.0,>=0.25.0->weaviate-client) (1.3.1)
Requirement already satisfied: h11<0.15,>=0.13 in
/usr/local/lib/python3.10/dist-packages (from
httpcore==1.*->httpx<=0.27.0,>=0.25.0->weaviate-client) (0.14.0)
Requirement already satisfied: annotated-types>=0.6.0 in
/usr/local/lib/python3.10/dist-packages (from pydantic<3.0.0,>=2.5.0->weaviate-
client) (0.7.0)
Requirement already satisfied: pydantic-core==2.23.4 in
/usr/local/lib/python3.10/dist-packages (from pydantic<3.0.0,>=2.5.0->weaviate-
client) (2.23.4)
Requirement already satisfied: typing-extensions>=4.6.1 in
/usr/local/lib/python3.10/dist-packages (from pydantic<3.0.0,>=2.5.0->weaviate-
client) (4.12.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.30.0->weaviate-
client) (3.4.0)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.30.0->weaviate-
client) (2.2.3)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-
packages (from anyio->httpx<=0.27.0,>=0.25.0->weaviate-client) (1.2.2)
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.10/dist-
packages (from cryptography->authlib<1.3.2,>=1.2.1->weaviate-client) (1.17.1)
Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist-
packages (from cffi>=1.12->cryptography->authlib<1.3.2,>=1.2.1->weaviate-client)
(2.22)
Downloading weaviate_client-4.9.0-py3-none-any.whl (378 kB)
                         378.2/378.2 kB
10.6 MB/s eta 0:00:00
Downloading validators-0.34.0-py3-none-any.whl (43 kB)
                         43.5/43.5 kB
2.7 MB/s eta 0:00:00
Downloading Authlib-1.3.1-py2.py3-none-any.whl (223 kB)
                         223.8/223.8 kB
14.5 MB/s eta 0:00:00
Downloading grpcio_health_checking-1.67.1-py3-none-any.whl (18 kB)
Downloading
grpcio-1.67.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (5.9
MB)
                         5.9/5.9 MB
57.6 MB/s eta 0:00:00
Downloading
grpcio_tools-1.67.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(2.4 MB)
                         2.4/2.4 MB
```

```
53.2 MB/s eta 0:00:00
Downloading httpx-0.27.0-py3-none-any.whl (75 kB)
                         75.6/75.6 kB
4.6 MB/s eta 0:00:00
Downloading protobuf-5.28.3-cp38-abi3-manylinux2014_x86_64.whl (316 kB)
                         316.6/316.6 kB
20.7 MB/s eta 0:00:00
Installing collected packages: validators, protobuf, grpcio, httpx,
grpcio-tools, grpcio-health-checking, authlib, weaviate-client
  Attempting uninstall: protobuf
    Found existing installation: protobuf 3.20.3
    Uninstalling protobuf-3.20.3:
      Successfully uninstalled protobuf-3.20.3
  Attempting uninstall: grpcio
    Found existing installation: grpcio 1.64.1
    Uninstalling grpcio-1.64.1:
      Successfully uninstalled grpcio-1.64.1
  Attempting uninstall: httpx
    Found existing installation: httpx 0.27.2
    Uninstalling httpx-0.27.2:
      Successfully uninstalled httpx-0.27.2
ERROR: pip's dependency resolver does not currently take into account all
the packages that are installed. This behaviour is the source of the following
dependency conflicts.
google-cloud-datastore 2.19.0 requires protobuf!=3.20.0,!=3.20.1,!=4.21.0,!=4.21
.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.19.5, but you have protobuf
5.28.3 which is incompatible.
google-cloud-firestore 2.16.1 requires protobuf!=3.20.0,!=3.20.1,!=4.21.0,!=4.21
.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.19.5, but you have protobuf
5.28.3 which is incompatible.
tensorboard 2.17.0 requires protobuf!=4.24.0,<5.0.0,>=3.19.6, but you have
protobuf 5.28.3 which is incompatible.
tensorflow 2.17.0 requires protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,
!=4.21.5,<5.0.0dev,>=3.20.3, but you have protobuf 5.28.3 which is incompatible.
tensorflow-metadata 1.16.1 requires protobuf<4.21,>=3.20.3; python_version <
"3.11", but you have protobuf 5.28.3 which is incompatible.
Successfully installed authlib-1.3.1 grpcio-1.67.1 grpcio-health-
checking-1.67.1 grpcio-tools-1.67.1 httpx-0.27.0 protobuf-5.28.3
validators-0.34.0 weaviate-client-4.9.0
```

[5]:
```
!pip install langchain
```

```
Requirement already satisfied: langchain in /usr/local/lib/python3.10/dist-
packages (0.3.4)
Requirement already satisfied: PyYAML>=5.3 in /usr/local/lib/python3.10/dist-
packages (from langchain) (6.0.2)
Requirement already satisfied: SQLAlchemy<3,>=1.4 in
/usr/local/lib/python3.10/dist-packages (from langchain) (2.0.36)
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in
/usr/local/lib/python3.10/dist-packages (from langchain) (3.10.10)
Requirement already satisfied: async-timeout<5.0.0,>=4.0.0 in
/usr/local/lib/python3.10/dist-packages (from langchain) (4.0.3)
Requirement already satisfied: langchain-core<0.4.0,>=0.3.12 in
/usr/local/lib/python3.10/dist-packages (from langchain) (0.3.13)
Requirement already satisfied: langchain-text-splitters<0.4.0,>=0.3.0 in
/usr/local/lib/python3.10/dist-packages (from langchain) (0.3.0)
Requirement already satisfied: langsmith<0.2.0,>=0.1.17 in
/usr/local/lib/python3.10/dist-packages (from langchain) (0.1.137)
Requirement already satisfied: numpy<2,>=1 in /usr/local/lib/python3.10/dist-
packages (from langchain) (1.26.4)
Requirement already satisfied: pydantic<3.0.0,>=2.7.4 in
/usr/local/lib/python3.10/dist-packages (from langchain) (2.9.2)
Requirement already satisfied: requests<3,>=2 in /usr/local/lib/python3.10/dist-
packages (from langchain) (2.32.3)
Requirement already satisfied: tenacity!=8.4.0,<10,>=8.1.0 in
/usr/local/lib/python3.10/dist-packages (from langchain) (9.0.0)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain)
(2.4.3)
Requirement already satisfied: aiosignal>=1.1.2 in
/usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain)
(1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-
packages (from aiohttp<4.0.0,>=3.8.3->langchain) (24.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain)
(1.5.0)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain)
(6.1.0)
Requirement already satisfied: yarl<2.0,>=1.12.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain)
(1.16.0)
Requirement already satisfied: jsonpatch<2.0,>=1.33 in
/usr/local/lib/python3.10/dist-packages (from langchain-
core<0.4.0,>=0.3.12->langchain) (1.33)
Requirement already satisfied: packaging<25,>=23.2 in
/usr/local/lib/python3.10/dist-packages (from langchain-
```

```
core<0.4.0,>=0.3.12->langchain) (24.1)
Requirement already satisfied: typing-extensions>=4.7 in
/usr/local/lib/python3.10/dist-packages (from langchain-
core<0.4.0,>=0.3.12->langchain) (4.12.2)
Requirement already satisfied: httpx<1,>=0.23.0 in
/usr/local/lib/python3.10/dist-packages (from
langsmith<0.2.0,>=0.1.17->langchain) (0.27.0)
Requirement already satisfied: orjson<4.0.0,>=3.9.14 in
/usr/local/lib/python3.10/dist-packages (from
langsmith<0.2.0,>=0.1.17->langchain) (3.10.10)
Requirement already satisfied: requests-toolbelt<2.0.0,>=1.0.0 in
/usr/local/lib/python3.10/dist-packages (from
langsmith<0.2.0,>=0.1.17->langchain) (1.0.0)
Requirement already satisfied: annotated-types>=0.6.0 in
/usr/local/lib/python3.10/dist-packages (from pydantic<3.0.0,>=2.7.4->langchain)
(0.7.0)
Requirement already satisfied: pydantic-core==2.23.4 in
/usr/local/lib/python3.10/dist-packages (from pydantic<3.0.0,>=2.7.4->langchain)
(2.23.4)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2->langchain) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests<3,>=2->langchain) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2->langchain) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2->langchain)
(2024.8.30)
Requirement already satisfied: greenlet!=0.4.17 in
/usr/local/lib/python3.10/dist-packages (from SQLAlchemy<3,>=1.4->langchain)
(3.1.1)
Requirement already satisfied: anyio in /usr/local/lib/python3.10/dist-packages
(from httpx<1,>=0.23.0->langsmith<0.2.0,>=0.1.17->langchain) (3.7.1)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.10/dist-
packages (from httpx<1,>=0.23.0->langsmith<0.2.0,>=0.1.17->langchain) (1.0.6)
Requirement already satisfied: sniffio in /usr/local/lib/python3.10/dist-
packages (from httpx<1,>=0.23.0->langsmith<0.2.0,>=0.1.17->langchain) (1.3.1)
Requirement already satisfied: h11<0.15,>=0.13 in
/usr/local/lib/python3.10/dist-packages (from
httpcore==1.*->httpx<1,>=0.23.0->langsmith<0.2.0,>=0.1.17->langchain) (0.14.0)
Requirement already satisfied: jsonpointer>=1.9 in
/usr/local/lib/python3.10/dist-packages (from jsonpatch<2.0,>=1.33->langchain-
core<0.4.0,>=0.3.12->langchain) (3.0.0)
Requirement already satisfied: propcache>=0.2.0 in
/usr/local/lib/python3.10/dist-packages (from
yarl<2.0,>=1.12.0->aiohttp<4.0.0,>=3.8.3->langchain) (0.2.0)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-
packages (from anyio->httpx<1,>=0.23.0->langsmith<0.2.0,>=0.1.17->langchain)
```

(1.2.2)

[16]:
```
!pip install pypdf
```

```
Collecting pypdf
  Downloading pypdf-5.1.0-py3-none-any.whl.metadata (7.2 kB)
Requirement already satisfied: typing_extensions>=4.0 in
/usr/local/lib/python3.10/dist-packages (from pypdf) (4.12.2)
Downloading pypdf-5.1.0-py3-none-any.whl (297 kB)
                         0.0/298.0 kB
? eta -:--:--
                      297.0/298.0

kB 11.9 MB/s eta 0:00:01
                      298.0/298.0 kB
7.6 MB/s eta 0:00:00
Installing collected packages: pypdf
Successfully installed pypdf-5.1.0
```

[17]:
```
!pip install tiktoken
```

```
Collecting tiktoken
  Downloading tiktoken-0.8.0-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (6.6 kB)
Requirement already satisfied: regex>=2022.1.18 in
/usr/local/lib/python3.10/dist-packages (from tiktoken) (2024.9.11)
Requirement already satisfied: requests>=2.26.0 in
/usr/local/lib/python3.10/dist-packages (from tiktoken) (2.32.3)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.26.0->tiktoken)
(3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests>=2.26.0->tiktoken) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.26.0->tiktoken)
(2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.26.0->tiktoken)
(2024.8.30)
Downloading
tiktoken-0.8.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.2
MB)
                         0.0/1.2 MB
? eta -:--:--
                      0.2/1.2

MB 7.5 MB/s eta 0:00:01
                      1.2/1.2 MB
24.0 MB/s eta 0:00:01
```

```
                         1.2/1.2 MB 16.1
MB/s eta 0:00:00
Installing collected packages: tiktoken
Successfully installed tiktoken-0.8.0
```

[18]:
```
!pip install rapidocr-onnxruntime
```

```
Collecting rapidocr-onnxruntime
  Downloading rapidocr_onnxruntime-1.3.25-py3-none-any.whl.metadata (1.3 kB)
Collecting pyclipper>=1.2.0 (from rapidocr-onnxruntime)
  Downloading pyclipper-1.3.0.post6-cp310-cp310-
manylinux_2_12_x86_64.manylinux2010_x86_64.whl.metadata (9.0 kB)
Requirement already satisfied: opencv-python>=4.5.1.48 in
/usr/local/lib/python3.10/dist-packages (from rapidocr-onnxruntime) (4.10.0.84)
Requirement already satisfied: numpy<3.0.0,>=1.19.5 in
/usr/local/lib/python3.10/dist-packages (from rapidocr-onnxruntime) (1.26.4)
Requirement already satisfied: six>=1.15.0 in /usr/local/lib/python3.10/dist-
packages (from rapidocr-onnxruntime) (1.16.0)
Requirement already satisfied: Shapely!=2.0.4,>=1.7.1 in
/usr/local/lib/python3.10/dist-packages (from rapidocr-onnxruntime) (2.0.6)
Requirement already satisfied: PyYAML in /usr/local/lib/python3.10/dist-packages
(from rapidocr-onnxruntime) (6.0.2)
Requirement already satisfied: Pillow in /usr/local/lib/python3.10/dist-packages
(from rapidocr-onnxruntime) (10.4.0)
Collecting onnxruntime>=1.7.0 (from rapidocr-onnxruntime)
  Downloading onnxruntime-1.19.2-cp310-cp310-
manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl.metadata (4.5 kB)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages
(from rapidocr-onnxruntime) (4.66.5)
Collecting coloredlogs (from onnxruntime>=1.7.0->rapidocr-onnxruntime)
  Downloading coloredlogs-15.0.1-py2.py3-none-any.whl.metadata (12 kB)
Requirement already satisfied: flatbuffers in /usr/local/lib/python3.10/dist-
packages (from onnxruntime>=1.7.0->rapidocr-onnxruntime) (24.3.25)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-
packages (from onnxruntime>=1.7.0->rapidocr-onnxruntime) (24.1)
Requirement already satisfied: protobuf in /usr/local/lib/python3.10/dist-
packages (from onnxruntime>=1.7.0->rapidocr-onnxruntime) (5.28.3)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages
(from onnxruntime>=1.7.0->rapidocr-onnxruntime) (1.13.1)
Collecting humanfriendly>=9.1 (from coloredlogs->onnxruntime>=1.7.0->rapidocr-
onnxruntime)
  Downloading humanfriendly-10.0-py2.py3-none-any.whl.metadata (9.2 kB)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from
sympy->onnxruntime>=1.7.0->rapidocr-onnxruntime) (1.3.0)
Downloading rapidocr_onnxruntime-1.3.25-py3-none-any.whl (14.9 MB)
                            14.9/14.9 MB
```

```
58.5 MB/s eta 0:00:00
Downloading
onnxruntime-1.19.2-cp310-cp310-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl
(13.2 MB)
                         13.2/13.2 MB
52.3 MB/s eta 0:00:00
Downloading
pyclipper-1.3.0.post6-cp310-cp310-manylinux_2_12_x86_64.manylinux2010_x86_64.whl
(912 kB)
                         912.2/912.2 kB
34.2 MB/s eta 0:00:00
Downloading coloredlogs-15.0.1-py2.py3-none-any.whl (46 kB)
                         46.0/46.0 kB
3.5 MB/s eta 0:00:00
Downloading humanfriendly-10.0-py2.py3-none-any.whl (86 kB)
                         86.8/86.8 kB
6.6 MB/s eta 0:00:00
Installing collected packages: pyclipper, humanfriendly, coloredlogs,
onnxruntime, rapidocr-onnxruntime
Successfully installed coloredlogs-15.0.1 humanfriendly-10.0 onnxruntime-1.19.2
pyclipper-1.3.0.post6 rapidocr-onnxruntime-1.3.25
```

[9]:

```python
import os
import weaviate # importing weaviate

# Best practice: store your credentials in environment variables
# WEAVIATE_URL       your Weaviate instance URL
# WEAVIATE_API_KEY   your Weaviate instance API key

# Get the URL and API key from environment variables.
# Use default values if not set.
WEAVIATE_URL = os.getenv("WEAVIATE_URL", "https://gbly7hkftw2fzmzrv53szw.c0.
↪europe-west3.gcp.weaviate.cloud")  # Replace with your actual Weaviate URL
WEAVIATE_API_KEY = os.getenv("WEAVIATE_API_KEY",␣
↪"9OJ0IWuCDsgxHHLoF2rpEV8kjfjhDDZK3cMN")  # Replace with your actual Weaviate␣
↪API key


# Create the client
client = weaviate.Client(
    url=WEAVIATE_URL,
    auth_client_secret=weaviate.auth.AuthApiKey(api_key=WEAVIATE_API_KEY),
)

print("\nIs client ready???", client.is_ready())
```

```
<ipython-input-9-2ee3f77acc46>:15: DeprecationWarning:
Python client v3 `weaviate.Client(…)` connections and methods are deprecated
```

```
and will
            be removed by 2024-11-30.

            Upgrade your code to use Python client v4 `weaviate.WeaviateClient`
connections and methods.
                - For Python Client v4 usage, see:
https://weaviate.io/developers/weaviate/client-libraries/python
                - For code migration, see:
https://weaviate.io/developers/weaviate/client-libraries/python/v3_v4_migration

            If you have to use v3 code, install the v3 client and pin the v3
dependency in your requirements file: `weaviate-client>=3.26.7;<4.0.0`
  client = weaviate.Client(


Is client ready??? True
```

[10]:

```python
# These lines of code will fix uni-code error
import locale
locale.getpreferredencoding = lambda: "UTF-8"
```

[13]:

```python
# her importing HuggingFaceEmbeddings
from langchain.embeddings import HuggingFaceEmbeddings

# model name which is going to use
embeddings_model_name = 'sentence-transformers/all-mpnet-base-v2'
embeddings = HuggingFaceEmbeddings(model_name=embeddings_model_name)
```

```
<ipython-input-13-32c6ea10de14>:6: LangChainDeprecationWarning: The class
`HuggingFaceEmbeddings` was deprecated in LangChain 0.2.2 and will be removed in
1.0. An updated version of the class exists in the :class:`~langchain-
huggingface package and should be used instead. To use it run `pip install -U
:class:`~langchain-huggingface` and import as `from
:class:`~langchain_huggingface import HuggingFaceEmbeddings``.
  embeddings = HuggingFaceEmbeddings(model_name=embeddings_model_name)
/usr/local/lib/python3.10/dist-
packages/sentence_transformers/cross_encoder/CrossEncoder.py:13:
TqdmExperimentalWarning: Using `tqdm.autonotebook.tqdm` in notebook mode. Use
`tqdm.tqdm` instead to force console mode (e.g. in jupyter console)
  from tqdm.autonotebook import tqdm, trange
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89:
UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab
(https://huggingface.co/settings/tokens), set it as secret in your Google Colab
and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access
public models or datasets.
```

```
  warnings.warn(

modules.json:   0%|          | 0.00/349 [00:00<?, ?B/s]

config_sentence_transformers.json:   0%|          | 0.00/116 [00:00<?, ?B/s]

README.md:   0%|          | 0.00/10.6k [00:00<?, ?B/s]

sentence_bert_config.json:   0%|          | 0.00/53.0 [00:00<?, ?B/s]

config.json:   0%|          | 0.00/571 [00:00<?, ?B/s]

model.safetensors:   0%|          | 0.00/438M [00:00<?, ?B/s]

tokenizer_config.json:   0%|          | 0.00/363 [00:00<?, ?B/s]

vocab.txt:   0%|          | 0.00/232k [00:00<?, ?B/s]

tokenizer.json:   0%|          | 0.00/466k [00:00<?, ?B/s]

special_tokens_map.json:   0%|          | 0.00/239 [00:00<?, ?B/s]

/usr/local/lib/python3.10/dist-
packages/transformers/tokenization_utils_base.py:1601: FutureWarning:
`clean_up_tokenization_spaces` was not set. It will be set to `True` by default.
This behavior will be depracted in transformers v4.45, and will be then set to
`False` by default. For more details check this issue:
https://github.com/huggingface/transformers/issues/31884
  warnings.warn(

1_Pooling/config.json:   0%|          | 0.00/190 [00:00<?, ?B/s]
```

[12]:

```
!pip install -U langchain-community
```

```
Collecting langchain-community
  Downloading langchain_community-0.3.4-py3-none-any.whl.metadata (2.9 kB)
Requirement already satisfied: PyYAML>=5.3 in /usr/local/lib/python3.10/dist-
packages (from langchain-community) (6.0.2)
Requirement already satisfied: SQLAlchemy<3,>=1.4 in
/usr/local/lib/python3.10/dist-packages (from langchain-community) (2.0.36)
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in
/usr/local/lib/python3.10/dist-packages (from langchain-community) (3.10.10)
Collecting dataclasses-json<0.7,>=0.5.7 (from langchain-community)
  Downloading dataclasses_json-0.6.7-py3-none-any.whl.metadata (25 kB)
Collecting httpx-sse<0.5.0,>=0.4.0 (from langchain-community)
  Downloading httpx_sse-0.4.0-py3-none-any.whl.metadata (9.0 kB)
Collecting langchain<0.4.0,>=0.3.6 (from langchain-community)
  Downloading langchain-0.3.6-py3-none-any.whl.metadata (7.1 kB)
Collecting langchain-core<0.4.0,>=0.3.14 (from langchain-community)
  Downloading langchain_core-0.3.14-py3-none-any.whl.metadata (6.3 kB)
Requirement already satisfied: langsmith<0.2.0,>=0.1.125 in
/usr/local/lib/python3.10/dist-packages (from langchain-community) (0.1.137)
Requirement already satisfied: numpy<2,>=1 in /usr/local/lib/python3.10/dist-
packages (from langchain-community) (1.26.4)
```

```
Collecting pydantic-settings<3.0.0,>=2.4.0 (from langchain-community)
  Downloading pydantic_settings-2.6.0-py3-none-any.whl.metadata (3.5 kB)
Requirement already satisfied: requests<3,>=2 in /usr/local/lib/python3.10/dist-
packages (from langchain-community) (2.32.3)
Requirement already satisfied: tenacity!=8.4.0,<10,>=8.1.0 in
/usr/local/lib/python3.10/dist-packages (from langchain-community) (9.0.0)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain-
community) (2.4.3)
Requirement already satisfied: aiosignal>=1.1.2 in
/usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain-
community) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-
packages (from aiohttp<4.0.0,>=3.8.3->langchain-community) (24.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain-
community) (1.5.0)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain-
community) (6.1.0)
Requirement already satisfied: yarl<2.0,>=1.12.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain-
community) (1.16.0)
Requirement already satisfied: async-timeout<5.0,>=4.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain-
community) (4.0.3)
Collecting marshmallow<4.0.0,>=3.18.0 (from dataclasses-
json<0.7,>=0.5.7->langchain-community)
  Downloading marshmallow-3.23.0-py3-none-any.whl.metadata (7.6 kB)
Collecting typing-inspect<1,>=0.4.0 (from dataclasses-
json<0.7,>=0.5.7->langchain-community)
  Downloading typing_inspect-0.9.0-py3-none-any.whl.metadata (1.5 kB)
Requirement already satisfied: langchain-text-splitters<0.4.0,>=0.3.0 in
/usr/local/lib/python3.10/dist-packages (from
langchain<0.4.0,>=0.3.6->langchain-community) (0.3.0)
Requirement already satisfied: pydantic<3.0.0,>=2.7.4 in
/usr/local/lib/python3.10/dist-packages (from
langchain<0.4.0,>=0.3.6->langchain-community) (2.9.2)
Requirement already satisfied: jsonpatch<2.0,>=1.33 in
/usr/local/lib/python3.10/dist-packages (from langchain-
core<0.4.0,>=0.3.14->langchain-community) (1.33)
Requirement already satisfied: packaging<25,>=23.2 in
/usr/local/lib/python3.10/dist-packages (from langchain-
core<0.4.0,>=0.3.14->langchain-community) (24.1)
Requirement already satisfied: typing-extensions>=4.7 in
/usr/local/lib/python3.10/dist-packages (from langchain-
core<0.4.0,>=0.3.14->langchain-community) (4.12.2)
Requirement already satisfied: httpx<1,>=0.23.0 in
```

```
/usr/local/lib/python3.10/dist-packages (from
langsmith<0.2.0,>=0.1.125->langchain-community) (0.27.0)
Requirement already satisfied: orjson<4.0.0,>=3.9.14 in
/usr/local/lib/python3.10/dist-packages (from
langsmith<0.2.0,>=0.1.125->langchain-community) (3.10.10)
Requirement already satisfied: requests-toolbelt<2.0.0,>=1.0.0 in
/usr/local/lib/python3.10/dist-packages (from
langsmith<0.2.0,>=0.1.125->langchain-community) (1.0.0)
Collecting python-dotenv>=0.21.0 (from pydantic-
settings<3.0.0,>=2.4.0->langchain-community)
  Downloading python_dotenv-1.0.1-py3-none-any.whl.metadata (23 kB)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2->langchain-
community) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests<3,>=2->langchain-community) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2->langchain-
community) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2->langchain-
community) (2024.8.30)
Requirement already satisfied: greenlet!=0.4.17 in
/usr/local/lib/python3.10/dist-packages (from SQLAlchemy<3,>=1.4->langchain-
community) (3.1.1)
Requirement already satisfied: anyio in /usr/local/lib/python3.10/dist-packages
(from httpx<1,>=0.23.0->langsmith<0.2.0,>=0.1.125->langchain-community) (3.7.1)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.10/dist-
packages (from httpx<1,>=0.23.0->langsmith<0.2.0,>=0.1.125->langchain-community)
(1.0.6)
Requirement already satisfied: sniffio in /usr/local/lib/python3.10/dist-
packages (from httpx<1,>=0.23.0->langsmith<0.2.0,>=0.1.125->langchain-community)
(1.3.1)
Requirement already satisfied: h11<0.15,>=0.13 in
/usr/local/lib/python3.10/dist-packages (from
httpcore==1.*->httpx<1,>=0.23.0->langsmith<0.2.0,>=0.1.125->langchain-community)
(0.14.0)
Requirement already satisfied: jsonpointer>=1.9 in
/usr/local/lib/python3.10/dist-packages (from jsonpatch<2.0,>=1.33->langchain-
core<0.4.0,>=0.3.14->langchain-community) (3.0.0)
Requirement already satisfied: annotated-types>=0.6.0 in
/usr/local/lib/python3.10/dist-packages (from
pydantic<3.0.0,>=2.7.4->langchain<0.4.0,>=0.3.6->langchain-community) (0.7.0)
Requirement already satisfied: pydantic-core==2.23.4 in
/usr/local/lib/python3.10/dist-packages (from
pydantic<3.0.0,>=2.7.4->langchain<0.4.0,>=0.3.6->langchain-community) (2.23.4)
Collecting mypy-extensions>=0.3.0 (from typing-inspect<1,>=0.4.0->dataclasses-
json<0.7,>=0.5.7->langchain-community)
```

```
  Downloading mypy_extensions-1.0.0-py3-none-any.whl.metadata (1.1 kB)
Requirement already satisfied: propcache>=0.2.0 in
/usr/local/lib/python3.10/dist-packages (from
yarl<2.0,>=1.12.0->aiohttp<4.0.0,>=3.8.3->langchain-community) (0.2.0)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-
packages (from anyio->httpx<1,>=0.23.0->langsmith<0.2.0,>=0.1.125->langchain-
community) (1.2.2)
Downloading langchain_community-0.3.4-py3-none-any.whl (2.4 MB)
                         2.4/2.4 MB
39.4 MB/s eta 0:00:00
Downloading dataclasses_json-0.6.7-py3-none-any.whl (28 kB)
Downloading httpx_sse-0.4.0-py3-none-any.whl (7.8 kB)
Downloading langchain-0.3.6-py3-none-any.whl (1.0 MB)
                         1.0/1.0 MB
35.4 MB/s eta 0:00:00
Downloading langchain_core-0.3.14-py3-none-any.whl (408 kB)
                         408.7/408.7 kB
23.5 MB/s eta 0:00:00
Downloading pydantic_settings-2.6.0-py3-none-any.whl (28 kB)
Downloading marshmallow-3.23.0-py3-none-any.whl (49 kB)
                         49.5/49.5 kB
3.6 MB/s eta 0:00:00
Downloading python_dotenv-1.0.1-py3-none-any.whl (19 kB)
Downloading typing_inspect-0.9.0-py3-none-any.whl (8.8 kB)
Downloading mypy_extensions-1.0.0-py3-none-any.whl (4.7 kB)
Installing collected packages: python-dotenv, mypy-extensions, marshmallow,
httpx-sse, typing-inspect, pydantic-settings, dataclasses-json, langchain-core,
langchain, langchain-community
  Attempting uninstall: langchain-core
    Found existing installation: langchain-core 0.3.13
    Uninstalling langchain-core-0.3.13:
      Successfully uninstalled langchain-core-0.3.13
  Attempting uninstall: langchain
    Found existing installation: langchain 0.3.4
    Uninstalling langchain-0.3.4:
      Successfully uninstalled langchain-0.3.4
Successfully installed dataclasses-json-0.6.7 httpx-sse-0.4.0 langchain-0.3.6
langchain-community-0.3.4 langchain-core-0.3.14 marshmallow-3.23.0 mypy-
extensions-1.0.0 pydantic-settings-2.6.0 python-dotenv-1.0.1 typing-
inspect-0.9.0
```

[20]:

```python
# PyPDF to work with pdf files
from langchain.document_loaders import PyPDFLoader

# path of the pdf files
path = "/content/Retrieval-Augmented Generation for NLP Tasks.pdf"
```

```python
#loading file
loader = PyPDFLoader(path, extract_images=True)
pages = loader.load()
```

[21]:
```python
pages
```

[21]:

```
[Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 0}, page_content='Retrieval-Augmented Generation
for\nKnowledge-Intensive NLP Tasks\nPatrick Lewis†‡, Ethan Perez ,\nAleksandra
Piktus†, Fabio Petroni†, Vladimir Karpukhin†, Naman Goyal†, Heinrich
Küttler†,\nMike Lewis†, Wen-tau Yih†, Tim Rocktäschel†‡, Sebastian Riedel†‡,
Douwe Kiela†\n†Facebook AI Research; ‡University College London;  New York
University;\nplewis@fb.com\nAbstract\nLarge pre-trained language models have
been shown to store factual knowledge\nin their parameters, and achieve state-
of-the-art results when  ne-tuned on down-\nstream NLP tasks. However, their
ability to access and precisely manipulate knowl-\nedge is still limited, and
hence on knowledge-intensive tasks, their performance\nlags behind task-speci c
architectures. Additionally, providing provenance for their\ndecisions and
updating their world knowledge remain open research problems. Pre-\ntrained
models with a differentiable access mechanism to explicit non-parametric\nmemory
have so far been only investigated for extractive downstream tasks. We\nexplore
a general-purpose  ne-tuning recipe for retrieval-augmented generation\n(RAG) -
models which combine pre-trained parametric and non-parametric mem-\nory for
language generation. We introduce RAG models where the parametric\nmemory is a
pre-trained seq2seq model and the non-parametric memory is a dense\nvector index
of Wikipedia, accessed with a pre-trained neural retriever. We com-\npare two
RAG formulations, one which conditions on the same retrieved passages\nacross
the whole generated sequence, and another which can use different passages\nper
token. We  ne-tune and evaluate our models on a wide range of
knowledge-\nintensive NLP tasks and set the state of the art on three open
domain QA tasks,\noutperforming parametric seq2seq models and task-speci c
retrieve-and-extract\narchitectures. For language generation tasks, we  nd that
RAG models generate\nmore speci c, diverse and factual language than a state-of-
the-art parametric-only\nseq2seq baseline.\n1 Introduction\nPre-trained neural
language models have been shown to learn a substantial amount of in-depth
knowl-\nedge from data [47]. They can do so without any access to an external
memory, as a parameterized\nimplicit knowledge base [51, 52]. While this
development is exciting, such models do have down-\nsides: They cannot easily
expand or revise their memory, can’t straightforwardly provide insight
into\ntheir predictions, and may produce "hallucinations" [38]. Hybrid models
that combine parametric\nmemory with non-parametric (i.e., retrieval-based)
memories [20, 26, 48] can address some of these\nissues because knowledge can be
directly revised and expanded, and accessed knowledge can be\ninspected and
interpreted. REALM [ 20] and ORQA [ 31], two recently introduced models
that\ncombine masked language models [8] with a differentiable retriever, have
shown promising results,\narXiv:2005.11401v4  [cs.CL]  12 Apr 2021'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
```

Tasks.pdf', 'page': 1}, page_content='The\tDivine\nComedy\t(x) q \nQuery
\nEncoder \nq(x) \nMIPS p \nGenerator\xa0p \n(Parametric) \nMargin- \nalize \n
This\t14th\tcentury\twork\nis\tdivided\tinto\t3\nsections:\t"Inferno",\n"Purgato
rio"\t&\n"Paradiso"\t\t\t\t\t\t\t\t\t(y)\nEnd-to-End Backprop through q
and\xa0p \nBarack\tObama\twas\nborn\tin\tHawaii.(x)\nFact Veri cation: Fact
Query\nsupports\t(y)\nQuestion Generation\nFact Veri cation:\nLabel
Generation\nDocument \nIndex \nDefine\t"middle\tear"(x)\nQuestion
Answering:\nQuestion Query\nThe\tmiddle\tear\tincludes\nthe\ttympanic\tcavity\ta
nd\nthe\tthree\tossicles.\t\t(y)\nQuestion Answering:\nAnswer
GenerationRetriever p \n(Non-Parametric) \nz 4 \nz 3 \nz 2 \nz 1 \nd(z)
\nJeopardy Question\nGeneration:\nAnswer Query\nFigure 1: Overview of our
approach. We combine a pre-trained retriever (Query Encoder + Document\nIndex)
with a pre-trained seq2seq model (Generator) and ne-tune end-to-end. For query
x , we use\nMaximum Inner Product Search (MIPS) to nd the top-K documents z i .
For nal prediction y , we\ntreat z as a latent variable and marginalize over
seq2seq predictions given different documents.\nbut have only explored open-
domain extractive question answering. Here, we bring hybrid parametric\nand non-
parametric memory to the "workhorse of NLP," i.e. sequence-to-sequence (seq2seq)
models.\nWe endow pre-trained, parametric-memory generation models with a non-
parametric memory through\na general-purpose ne-tuning approach which we refer
to as retrieval-augmented generation (RAG).\nWe build RAG models where the
parametric memory is a pre-trained seq2seq transformer, and the\nnon-parametric
memory is a dense vector index of Wikipedia, accessed with a pre-trained
neural\nretriever. We combine these components in a probabilistic model trained
end-to-end (Fig. 1). The\nretriever (Dense Passage Retriever [26], henceforth
DPR) provides latent documents conditioned on\nthe input, and the seq2seq model
(BART [32]) then conditions on these latent documents together with\nthe input
to generate the output. We marginalize the latent documents with a top-K
approximation,\neither on a per-output basis (assuming the same document is
responsible for all tokens) or a per-token\nbasis (where different documents are
responsible for different tokens). Like T5 [51] or BART, RAG\ncan be ne-tuned
on any seq2seq task, whereby both the generator and retriever are jointly
learned.\nThere has been extensive previous work proposing architectures to
enrich systems with non-parametric\nmemory which are trained from scratch for
speci c tasks, e.g. memory networks [64, 55], stack-\naugmented networks [25]
and memory layers [30]. In contrast, we explore a setting where
both\nparametric and non-parametric memory components are pre-trained and pre-
loaded with extensive\nknowledge. Crucially, by using pre-trained access
mechanisms, the ability to access knowledge is\npresent without additional
training.\nOur results highlight the bene ts of combining parametric and non-
parametric memory with genera-\ntion for knowledge-intensive tasks-tasks that
humans could not reasonably be expected to perform\nwithout access to an
external knowledge source. Our RAG models achieve state-of-the-art results\non
open Natural Questions [29], WebQuestions [3] and CuratedTrec [2] and strongly
outperform\nrecent approaches that use specialised pre-training objectives on
TriviaQA [24]. Despite these being\nextractive tasks, we nd that unconstrained
generation outperforms previous extractive approaches.\nFor knowledge-intensive

generation, we experiment with MS-MARCO [1] and Jeopardy question\ngeneration,
and we nd that our models generate responses that are more factual, speci c,
and\ndiverse than a BART baseline. For FEVER [56] fact veri cation, we achieve
results within 4.3% of\nstate-of-the-art pipeline models which use strong
retrieval supervision. Finally, we demonstrate that\nthe non-parametric memory
can be replaced to update the models' knowledge as the world changes.1\n2
Methods\nWe explore RAG models, which use the input sequencex to retrieve text
documents z and use them\nas additional context when generating the target
sequence y . As shown in Figure 1, our models\nleverage two components: (i) a
retriever p (z |x) with parameters   that returns (top-K
truncated)\ndistributions over text passages given a query x and (ii) a
generator p (y i |x,z,y 1:i -1) parametrized\n1Code to run experiments with RAG
has been open-sourced as part of the HuggingFace Transform-\ners Library [66]
and can be found at
https://github.com/huggingface/transformers/blob/master/\nexamples/rag/. An
interactive demo of RAG models can be found at https://huggingface.co/rag/\n2'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 2}, page_content='by that generates a current token based
on a context of the previous i-1 tokens y1:i-1, the original\ninput xand a
retrieved passage z.\nTo train the retriever and generator end-to-end, we treat
the retrieved document as a latent variable.\nWe propose two models that
marginalize over the latent documents in different ways to produce
a\ndistribution over generated text. In one approach, RAG-Sequence, the model
uses the same document\nto predict each target token. The second approach, RAG-
Token, can predict each target token based\non a different document. In the
following, we formally introduce both models and then describe the\np and p
components, as well as the training and decoding procedure.\n2.1 Models\nRAG-
Sequence Model The RAG-Sequence model uses the same retrieved document to
generate\nthe complete sequence. Technically, it treats the retrieved document
as a single latent variable that\nis marginalized to get the seq2seq probability
p(y|x) via a top-K approximation. Concretely, the\ntop K documents are retrieved
using the retriever, and the generator produces the output sequence\nprobability
for each document, which are then marginalized,\npRAG-Sequence(y|x)
 \n \nz top-k(p(·|x))\np (z|x)p (y|x,z)
=\n \nz top-k(p(·|x))\np (z|x)\nN \ni\np (yi|x,z,y 1:i-1)\nRAG-Token Model In
the RAG-Token model we can draw a different latent document for each\ntarget
token and marginalize accordingly. This allows the generator to choose content
from several\ndocuments when producing an answer. Concretely, the top K
documents are retrieved using the\nretriever, and then the generator produces a
distribution for the next output token for each document,\nbefore marginalizing,
and repeating the process with the following output token, Formally, we
de ne:\npRAG-Token(y|x) \nN \ni\n \nz top-k(p(·|x))\np (z|x)p (yi|x,z,y
1:i-1)\nFinally, we note that RAG can be used for sequence classi cation tasks
by considering the target class\nas a target sequence of length one, in which
case RAG-Sequence and RAG-Token are equivalent.\n2.2 Retriever: DPR\nThe
retrieval component p (z|x) is based on DPR [26]. DPR follows a bi-encoder
architecture:\np (z|x) exp\n(\nd(z) q(x)\n)\nd(z) =BERTd(z), q(x)

=BERTq(x)\nwhere d(z) is a dense representation of a document produced by a
BERTBASE document encoder [8],\nand q(x) a query representation produced by a
query encoder, also based on BERTBASE. Calculating\ntop-k(p (·|x)), the list of
kdocuments zwith highest prior probability p (z|x), is a Maximum Inner\nProduct
Search (MIPS) problem, which can be approximately solved in sub-linear time
[23]. We use\na pre-trained bi-encoder from DPR to initialize our retriever and
to build the document index. This\nretriever was trained to retrieve documents
which contain answers to TriviaQA [24] questions and\nNatural Questions [29]. We
refer to the document index as the non-parametric memory.\n2.3 Generator:
BART\nThe generator component p (yi|x,z,y 1:i−1) could be modelled using any
encoder-decoder. We use\nBART-large [32], a pre-trained seq2seq transformer [58]
with 400M parameters. To combine the input\nxwith the retrieved content zwhen
generating from BART, we simply concatenate them. BART was\npre-trained using a
denoising objective and a variety of different noising functions. It has
obtained\nstate-of-the-art results on a diverse set of generation tasks and
outperforms comparably-sized T5\nmodels [32]. We refer to the BART generator
parameters as the parametric memory henceforth.\n2.4 Training\nWe jointly train
the retriever and generator components without any direct supervision on
what\ndocument should be retrieved. Given a ne-tuning training corpus of
input/output pairs (xj,yj), we\n3'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 3}, page_content='minimize the negative marginal log-
likelihood of each target, \nj−log p(yj|xj) using stochastic\ngradient descent
with Adam [28]. Updating the document encoder BERTd during training is costly
as\nit requires the document index to be periodically updated as REALM does
during pre-training [20].\nWe do not nd this step necessary for strong
performance, and keep the document encoder (and\nindex) xed, only ne-tuning
the query encoder BERTq and the BART generator.\n2.5 Decoding\nAt test time,
RAG-Sequence and RAG-Token require different ways to approximatearg
maxyp(y|x).\nRAG-Token The RAG-Token model can be seen as a standard,
autoregressive seq2seq genera-\ntor with transition probability:
p \n (yi|x,y1:i−1) = \nz top-k(p(·|x)) p (zi|x)p (yi|x,zi,y1:i−1) To\ndecode,
we can plug p \n (yi|x,y1:i−1) into a standard beam decoder.\nRAG-Sequence For
RAG-Sequence, the likelihood p(y|x) does not break into a conventional
per-\ntoken likelihood, hence we cannot solve it with a single beam search.
Instead, we run beam search for\neach document z, scoring each hypothesis using
p (yi|x,z,y 1:i−1). This yields a set of hypotheses\nY, some of which may not
have appeared in the beams of all documents. To estimate the probability\nof an
hypothesis y we run an additional forward pass for each document z for which y
does not\nappear in the beam, multiply generator probability with p (z|x) and
then sum the probabilities across\nbeams for the marginals. We refer to this
decoding procedure as "Thorough Decoding." For longer\noutput sequences, |Y|can
become large, requiring many forward passes. For more ef cient decoding,\nwe can
make a further approximation that p (y|x,zi) 0 where ywas not generated during
beam\nsearch from x,zi. This avoids the need to run additional forward passes
once the candidate set Y has\nbeen generated. We refer to this decoding
procedure as "Fast Decoding."\n3 Experiments\nWe experiment with RAG in a wide

range of knowledge-intensive tasks. For all experiments, we use\na single Wikipedia dump for our non-parametric knowledge source. Following Lee et al. [31] and\nKarpukhin et al. [26], we use the December 2018 dump. Each Wikipedia article is split into disjoint\n100-word chunks, to make a total of 21M documents. We use the document encoder to compute an\nembedding for each document, and build a single MIPS index using FAISS [23] with a Hierarchical\nNavigable Small World approximation for fast retrieval [37]. During training, we retrieve the top\nkdocuments for each query. We consider k {5,10}for training and set kfor test time using dev\ndata. We now discuss experimental details for each task.\n3.1 Open-domain Question Answering\nOpen-domain question answering (QA) is an important real-world application and common testbed\nfor knowledge-intensive tasks [20]. We treat questions and answers as input-output text pairs (x,y)\nand train RAG by directly minimizing the negative log-likelihood of answers. We compare RAG to\nthe popular extractive QA paradigm [5, 7, 31, 26], where answers are extracted spans from retrieved\ndocuments, relying primarily on non-parametric knowledge. We also compare to "Closed-Book\nQA" approaches [52], which, like RAG, generate answers, but which do not exploit retrieval, instead\nrelying purely on parametric knowledge. We consider four popular open-domain QA datasets: Natural\nQuestions (NQ) [29], TriviaQA (TQA) [24]. WebQuestions (WQ) [3] and CuratedTrec (CT) [2]. As\nCT and WQ are small, we follow DPR [26] by initializing CT and WQ models with our NQ RAG\nmodel. We use the same train/dev/test splits as prior work [31, 26] and report Exact Match (EM)\nscores. For TQA, to compare with T5 [52], we also evaluate on the TQA Wiki test set.\n3.2 Abstractive Question Answering\nRAG models can go beyond simple extractive QA and answer questions with free-form, abstractive\ntext generation. To test RAG’s natural language generation (NLG) in a knowledge-intensive setting,\nwe use the MSMARCO NLG task v2.1 [43]. The task consists of questions, ten gold passages\nretrieved from a search engine for each question, and a full sentence answer annotated from the\nretrieved passages. We do not use the supplied passages, only the questions and answers, to treat\n4'),

Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 4}, page_content='MSMARCO as an open-domain abstractive QA task. MSMARCO has some questions that cannot be\nanswered in a way that matches the reference answer without access to the gold passages, such as\n"What is the weather in V olcano, CA?" so performance will be lower without using gold passages.\nWe also note that some MSMARCO questions cannot be answered using Wikipedia alone. Here,\nRAG can rely on parametric knowledge to generate reasonable responses.\n3.3 Jeopardy Question Generation\nTo evaluate RAG’s generation abilities in a non-QA setting, we study open-domain question gen-\neration. Rather than use questions from standard open-domain QA tasks, which typically consist\nof short, simple questions, we propose the more demanding task of generating Jeopardy questions.\nJeopardy is an unusual format that consists of trying to guess an entity from a fact about that entity.\nFor example, "The World Cup" is the answer to the question "In 1986 Mexico scored as the rst\ncountry to host this international sports competition twice." As Jeopardy questions are precise,\nfactual statements, generating Jeopardy

questions conditioned on their answer entities constitutes a\nchallenging knowledge-intensive generation task.\nWe use the splits from SearchQA [10], with 100K train, 14K dev, and 27K test examples. As\nthis is a new task, we train a BART model for comparison. Following [67], we evaluate using the\nSQuAD-tuned Q-BLEU-1 metric [42]. Q-BLEU is a variant of BLEU with a higher weight for\nmatching entities and has higher correlation with human judgment for question generation than\nstandard metrics. We also perform two human evaluations, one to assess generation factuality, and\none for speci city. We de ne factuality as whether a statement can be corroborated by trusted external\nsources, and speci city as high mutual dependence between the input and output [33]. We follow\nbest practice and use pairwise comparative evaluation [34]. Evaluators are shown an answer and two\ngenerated questions, one from BART and one from RAG. They are then asked to pick one of four\noptions-quuestion A is better, question B is better, both are good, or neither is good.\n3.4 Fact Veri cation\nFEVER [56] requires classifying whether a natural language claim is supported or refuted by\nWikipedia, or whether there is not enough information to decide. The task requires retrieving\nevidence from Wikipedia relating to the claim and then reasoning over this evidence to classify\nwhether the claim is true, false, or unveri able from Wikipedia alone. FEVER is a retrieval problem\ncoupled with an challenging entailment reasoning task. It also provides an appropriate testbed for\nexploring the RAG models' ability to handle classi cation rather than generation. We map FEVER\nclass labels (supports, refutes, or not enough info) to single output tokens and directly train with\nclaim-class pairs. Crucially, unlike most other approaches to FEVER, we do not use supervision on\nretrieved evidence. In many real-world applications, retrieval supervision signals aren't available, and\nmodels that do not require such supervision will be applicable to a wider range of tasks. We explore\ntwo variants: the standard 3-way classi cation task (supports/refutes/not enough info) and the 2-way\n(supports/refutes) task studied in Thorne and Vlachos [57]. In both cases we report label accuracy.\n4 Results\n4.1 Open-domain Question Answering\nTable 1 shows results for RAG along with state-of-the-art models. On all four open-domain QA\ntasks, RAG sets a new state of the art (only on the T5-comparable split for TQA). RAG combines\nthe generation exibility of the "closed-book" (parametric only) approaches and the performance of\n"open-book" retrieval-based approaches. Unlike REALM and T5+SSM, RAG enjoys strong results\nwithout expensive, specialized "salient span masking" pre-training [20]. It is worth noting that RAG's\nretriever is initialized using DPR's retriever, which uses retrieval supervision on Natural Questions\nand TriviaQA. RAG compares favourably to the DPR QA system, which uses a BERT-based "cross-\nencoder" to re-rank documents, along with an extractive reader. RAG demonstrates that neither a\nre-ranker nor extractive reader is necessary for state-of-the-art performance.\nThere are several advantages to generating answers even when it is possible to extract them. Docu-\nments with clues about the answer but do not contain the answer verbatim can still contribute towards\na correct answer being generated, which is not possible with standard extractive approaches, leading\n5'),
Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP

Tasks.pdf', 'page': 5}, page_content='Table 1: Open-Domain QA Test Scores. For TQA,\nleft column uses the standard test set for Open-\nDomain QA, right column uses the TQA-Wiki\ntest set. See Appendix D for further details.\nModel NQ TQA WQ CT\nClosed\nBook\nT5-11B [52] 34.5 - /50.1 37.4 -\nT5-11B+SSM[52] 36.6 - /60.5 44.7 -\nOpen\nBook\nREALM [20] 40.4 - / - 40.7 46.8\nDPR [26] 41.5 57.9/ - 41.1 50.6\nRAG-Token 44.1 55.2/66.1 45.5 50.0\nRAG-Seq. 44.5 56.8/68.0 45.2 52.2\nTable 2: Generation and classi cation Test Scores.\nMS-MARCO SotA is [4], FEVER-3 is [68] and\nFEVER-2 is [57] *Uses gold context/evidence.\nBest model without gold access underlined.\nModel Jeopardy MSMARCO FVR3 FVR2\nB-1 QB-1 R-L B-1 Label Acc.\nSotA - - 49.8* 49.9* 76.8 92.2 *\nBART 15.1 19.7 38.2 41.6 64.0 81.1\nRAG-Tok. 17.3 22.2 40.1 41.5 72.5 89.5RAG-Seq. 14.7 21.4 40.8 44.2\nto more effective marginalization over documents. Furthermore, RAG can generate correct answers\neven when the correct answer is not in any retrieved document, achieving 11.8% accuracy in such\ncases for NQ, where an extractive model would score 0%.\n4.2 Abstractive Question Answering\nAs shown in Table 2, RAG-Sequence outperforms BART on Open MS-MARCO NLG by 2.6 Bleu\npoints and 2.6 Rouge-L points. RAG approaches state-of-the-art model performance, which is\nimpressive given that (i) those models access gold passages with speci c information required to\ngenerate the reference answer , (ii) many questions are unanswerable without the gold passages, and\n(iii) not all questions are answerable from Wikipedia alone. Table 3 shows some generated answers\nfrom our models. Qualitatively, we nd that RAG models hallucinate less and generate factually\ncorrect text more often than BART. Later, we also show that RAG generations are more diverse than\nBART generations (see §4.5).\n4.3 Jeopardy Question Generation\nTable 2 shows that RAG-Token performs better than RAG-Sequence on Jeopardy question generation,\nwith both models outperforming BART on Q-BLEU-1. 4 shows human evaluation results, over 452\npairs of generations from BART and RAG-Token. Evaluators indicated that BART was more factual\nthan RAG in only 7.1% of cases, while RAG was more factual in 42.7% of cases, and both RAG and\nBART were factual in a further 17% of cases, clearly demonstrating the effectiveness of RAG on\nthe task over a state-of-the-art generation model. Evaluators also nd RAG generations to be more\nspeci c by a large margin. Table 3 shows typical generations from each model.\nJeopardy questions often contain two separate pieces of information, and RAG-Token may perform\nbest because it can generate responses that combine content from several documents. Figure 2 shows\nan example. When generating "Sun", the posterior is high for document 2 which mentions "The\nSun Also Rises". Similarly, document 1 dominates the posterior when "A Farewell to Arms" is\ngenerated. Intriguingly, after the rst token of each book is generated, the document posterior attens.\nThis observation suggests that the generator can complete the titles without depending on speci c\ndocuments. In other words, the model's parametric knowledge is suf cient to complete the titles. We\n nd evidence for this hypothesis by feeding the BART-only baseline with the partial decoding"The\nSun. BART completes the generation "The Sun Also Rises" is a novel by this author of "The Sun\nAlso Rises" indicating the title "The Sun Also Rises" is stored in BART's parameters. Similarly,\nBART will complete the partial decoding "The Sun Also Rises" is a novel by this author of "A\nwith "The Sun Also Rises" is a

novel by this author of "A Farewell to Arms". This example shows\nhow parametric
and non-parametric memories work together-the non-parametric component helps\nto
guide the generation, drawing out speci c knowledge stored in the parametric
memory.\n4.4 Fact Veri cation\nTable 2 shows our results on FEVER. For 3-way
classi cation, RAG scores are within 4.3% of\nstate-of-the-art models, which are
complex pipeline systems with domain-speci c architectures and\nsubstantial
engineering, trained using intermediate retrieval supervision, which RAG does
not require.\n6'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 6}, page_content='Document 1: his works are considered
classics of American\nliterature … His wartime experiences formed the basis
for his novel\n"A Farewell to Arms"(1929) …\nDocument 2: … artists of the
1920s "Lost Generation" expatriate\ncommunity. His debut novel,"The Sun Also
Rises", was published\nin 1926.\nBOS\n"\nTheSunAlso\nR ises\n" is a\nnovel\nby
this\nauthor\nof " A\nFarewellto\nArms\n"\nDoc 1\nDoc 2\nDoc 3\nDoc 4\nDoc
5\nFigure 2: RAG-Token document posterior p(zi|x,yi,y−i) for each generated
token for input "Hem-\ningway" for Jeopardy generation with 5 retrieved
documents. The posterior for document 1 is high\nwhen generating "A Farewell to
Arms" and for document 2 when generating "The Sun Also Rises".\nTable 3:
Examples from generation tasks. RAG models generate more speci c and factually
accurate\nresponses. ‘?’ indicates factually incorrect responses, * indicates
partially correct responses.\nTask Input Model Generation\nMS-\nMARCO\nde ne
middle\near\nBART ?The middle ear is the part of the ear between the middle ear
and the nose.\nRAG-T The middle ear is the portion of the ear internal to the
eardrum.\nRAG-S The middle ear includes the tympanic cavity and the three
ossicles.\nwhat currency\nneeded in\nscotland\nBART The currency needed in
Scotland is Pound sterling.\nRAG-T Pound is the currency needed in
Scotland.\nRAG-S The currency needed in Scotland is the pound
sterling.\nJeopardy\nQuestion\nGener\n-ation\nWashington\nBART ?This state has
the largest number of counties in the U.S.\nRAG-T It’s the only U.S. state named
for a U.S. president\nRAG-S It’s the state where you’ll nd Mount Rainier
National Park\nThe Divine\nComedy\nBART *This epic poem by Dante is divided into
3 parts: the Inferno, the Purgatorio & the Purgatorio\nRAG-T Dante’s "Inferno"
is the rst part of this epic poem\nRAG-S This 14th century work is divided into
3 sections: "Inferno", "Purgatorio" & "Paradiso"\nFor 2-way classi cation, we
compare against Thorne and Vlachos [57], who train RoBERTa [35]\nto classify the
claim as true or false given the gold evidence sentence. RAG achieves an
accuracy\nwithin 2.7% of this model, despite being supplied with only the claim
and retrieving its own evidence.\nWe also analyze whether documents retrieved by
RAG correspond to documents annotated as gold\nevidence in FEVER. We calculate
the overlap in article titles between the topkdocuments retrieved\nby RAG and
gold evidence annotations. We nd that the top retrieved document is from a gold
article\nin 71% of cases, and a gold article is present in the top 10 retrieved
articles in 90% of cases.\n4.5 Additional Results\nGeneration Diversity Section
4.3 shows that RAG models are more factual and speci c than\nBART for Jeopardy
question generation. Following recent work on diversity-promoting decoding\n[33,
59, 39], we also investigate generation diversity by calculating the ratio of

distinct ngrams to\ntotal ngrams generated by different models. Table 5 shows that RAG-Sequence’s generations are\nmore diverse than RAG-Token’s, and both are signi cantly more diverse than BART without needing\nany diversity-promoting decoding.\nRetrieval Ablations A key feature of RAG is learning to retrieve relevant information for the task.\nTo assess the effectiveness of the retrieval mechanism, we run ablations where we freeze the retriever\nduring training. As shown in Table 6, learned retrieval improves results for all tasks.\nWe compare RAG’s dense retriever to a word overlap-based BM25 retriever [53]. Here, we replace\nRAG’s retriever with a xed BM25 system, and use BM25 retrieval scores as logits when calculating\np(z|x). Table 6 shows the results. For FEVER, BM25 performs best, perhaps since FEVER claims are\nheavily entity-centric and thus well-suited for word overlap-based retrieval. Differentiable retrieval\nimproves results on all other tasks, especially for Open-Domain QA, where it is crucial.\nIndex hot-swapping An advantage of non-parametric memory models like RAG is that knowledge\ncan be easily updated at test time. Parametric-only models like T5 or BART need further training to\nupdate their behavior as the world changes. To demonstrate, we build an index using the DrQA [5]\nWikipedia dump from December 2016 and compare outputs from RAG using this index to the newer\nindex from our main results (December 2018). We prepare a list of 82 world leaders who had changed\n7'),

Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 7}, page_content='Table 4: Human assessments for the Jeopardy\nQuestion Generation Task.\nFactuality Speci city\nBART better 7.1% 16.8%\nRAG better 42.7% 37.4%\nBoth good 11.7% 11.8%\nBoth poor 17.7% 6.9%\nNo majority 20.8% 20.1%\nTable 5: Ratio of distinct to total tri-grams for\ngeneration tasks.\nMSMARCO Jeopardy QGen\nGold 89.6% 90.0%\nBART 70.7% 32.4%\nRAG-Token 77.8% 46.8%\nRAG-Seq. 83.5% 53.8%\nTable 6: Ablations on the dev set. As FEVER is a classi cation task, both RAG models are equivalent.\nModel NQ TQA WQ CT Jeopardy-QGen MSMarco FVR-3 FVR-2\nExact Match B-1 QB-1 R-L B-1 Label Accuracy\nRAG-Token-BM25 29.7 41.5 32.1 33.1 17.5 22.3 55.5 48.4 75.1 91.6RAG-Sequence-BM25 31.8 44.1 36.6 33.8 11.1 19.5 56.5 46.9\nRAG-Token-Frozen 37.8 50.1 37.1 51.1 16.7 21.7 55.9 49.4 72.9 89.4RAG-Sequence-Frozen 41.2 52.1 41.8 52.6 11.8 19.6 56.7 47.3\nRAG-Token 43.5 54.8 46.5 51.9 17.9 22.6 56.2 49.4 74.5 90.6RAG-Sequence 44.0 55.8 44.9 53.4 15.3 21.5 57.2 47.5\nbetween these dates and use a template "Who is {position}?" (e.g. "Who is the President of Peru?")\nto query our NQ RAG model with each index. RAG answers 70% correctly using the 2016 index for\n2016 world leaders and 68% using the 2018 index for 2018 world leaders. Accuracy with mismatched\nindices is low (12% with the 2018 index and 2016 leaders, 4% with the 2016 index and 2018 leaders).\nThis shows we can update RAG’s world knowledge by simply replacing its non-parametric memory.\nEffect of Retrieving more documents Models are trained with either 5 or 10 retrieved latent\ndocuments, and we do not observe signi cant differences in performance between them. We have the\n exibility to adjust the number of retrieved documents at test time, which can affect performance and\nruntime. Figure 3 (left) shows that retrieving more documents at test time monotonically improves\nOpen-domain QA results for RAG-Sequence, but performance peaks for

RAG-Token at 10 retrieved\ndocuments. Figure 3 (right) shows that retrieving more documents leads to higher Rouge-L for\nRAG-Token at the expense of Bleu-1, but the effect is less pronounced for RAG-Sequence.\n10 20 30 40 50\nKR e t r i e v e dD o c s\n39\n40\n41\n42\n43\n44NQ Exact Match RAG-Tok\nRAG-Seq\n10 20 30 40 50\nKR e t r i e v e dD o c s\n40\n50\n60\n70\n80NQ Answer Recall @ K\nRAG-Tok\nRAG-Seq\nFixed DPR\nBM25\n10 20 30 40 50\nKR e t r i e v e dD o c s\n48\n50\n52\n54\n56Bleu-1 / Rouge-L score\nRAG-Tok R-L\nRAG-Tok B-1\nRAG-Seq R-L\nRAG-Seq B-1\nFigure 3: Left: NQ performance as more documents are retrieved. Center: Retrieval recall perfor-\nmance in NQ. Right: MS-MARCO Bleu-1 and Rouge-L as more documents are retrieved.\n5 Related Work\nSingle-Task Retrieval Prior work has shown that retrieval improves performance across a variety of\nNLP tasks when considered in isolation. Such tasks include open-domain question answering [5, 29],\nfact checking [56], fact completion [48], long-form question answering [12], Wikipedia article\ngeneration [36], dialogue [41, 65, 9, 13], translation [17], and language modeling [19, 27]. Our\nwork uni es previous successes in incorporating retrieval into individual tasks, showing that a single\nretrieval-based architecture is capable of achieving strong performance across several tasks.\n8'),

Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 8}, page_content='General-Purpose Architectures for NLP Prior work on general-purpose architectures for NLP\ntasks has shown great success without the use of retrieval. A single, pre-trained language model\nhas been shown to achieve strong performance on various classi cation tasks in the GLUE bench-\nmarks [60, 61] after ne-tuning [49, 8]. GPT-2 [50] later showed that a single, left-to-right, pre-trained\nlanguage model could achieve strong performance across both discriminative and generative tasks.\nFor further improvement, BART [32] and T5 [51, 52] propose a single, pre-trained encoder-decoder\nmodel that leverages bi-directional attention to achieve stronger performance on discriminative\nand generative tasks. Our work aims to expand the space of possible tasks with a single, uni ed\narchitecture, by learning a retrieval module to augment pre-trained, generative language models.\nLearned Retrieval There is signi cant work on learning to retrieve documents in information\nretrieval, more recently with pre-trained, neural language models [44, 26] similar to ours. Some\nwork optimizes the retrieval module to aid in a speci c, downstream task such as question answering,\nusing search [46], reinforcement learning [6, 63, 62], or a latent variable approach [31, 20] as in our\nwork. These successes leverage different retrieval-based architectures and optimization techniques to\nachieve strong performance on a single task, while we show that a single retrieval-based architecture\ncan be ne-tuned for strong performance on a variety of tasks.\nMemory-based Architectures Our document index can be seen as a large external memory for\nneural networks to attend to, analogous to memory networks [64, 55]. Concurrent work [14] learns\nto retrieve a trained embedding for each entity in the input, rather than to retrieve raw text as in our\nwork. Other work improves the ability of dialog models to generate factual text by attending over\nfact embeddings [15, 13]. A key feature of our memory is that it is comprised of raw text rather\ndistributed representations, which makes the memory both (i) human-readable, lending a form of\ninterpretability to our model, and (ii) human-writable, enabling us to dynamically update the model’s\nmemory by editing the document index. This approach has also been used in knowledge-intensive\ndialog, where generators have been conditioned on retrieved text directly, albeit obtained via TF-IDF\nrather than end-to-end learnt retrieval [9].\nRetrieve-and-Edit approaches Our method shares some similarities with retrieve-and-edit style\napproaches, where a similar training input-output pair is retrieved for a given input, and then edited\nto provide a nal output. These approaches have proved successful in a number of domains including\nMachine Translation [18, 22] and Semantic Parsing [21]. Our approach does have several differences,\nincluding less of emphasis on lightly editing a retrieved item, but on aggregating content from several\npieces of retrieved content, as well as learning latent retrieval, and retrieving evidence documents\nrather than related training pairs. This said, RAG techniques may work well in these settings, and\ncould represent promising future work.\n6 Discussion\nIn this work, we presented hybrid generation models with access to parametric and non-parametric\nmemory. We showed that our RAG models obtain state of the art results on open-domain QA. We\nfound that people prefer RAG’s generation over purely parametric BART, nding RAG more factual\nand speci c. We conducted an thorough investigation of the learned retrieval component, validating\nits effectiveness, and we illustrated how the retrieval index can be hot-swapped to update the model\nwithout requiring any retraining. In future work, it may be fruitful to investigate if the two components\ncan be jointly pre-trained from scratch, either with a denoising objective similar to BART or some\nanother objective. Our work opens up new research directions on how parametric and non-parametric\nmemories interact and how to most effectively combine them, showing promise in being applied to a\nwide variety of NLP tasks.\n9'),

Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 9}, page_content='Broader Impact\nThis work offers several positive societal bene ts over previous work: the fact that it is more\nstrongly grounded in real factual knowledge (in this case Wikipedia) makes it "hallucinate" less\nwith generations that are more factual, and offers more control and interpretability. RAG could be\nemployed in a wide variety of scenarios with direct bene t to society, for example by endowing it\nwith a medical index and asking it open-domain questions on that topic, or by helping people be more\neffective at their jobs.\nWith these advantages also come potential downsides: Wikipedia, or any potential external knowledge\nsource, will probably never be entirely factual and completely devoid of bias. Since RAG can be\nemployed as a language model, similar concerns as for GPT-2 [50] are valid here, although arguably\nto a lesser extent, including that it might be used to generate abuse, faked or misleading content in\nthe news or on social media; to impersonate others; or to automate the production of spam/phishing\ncontent [54]. Advanced language models may also lead to the automation of various jobs in the\ncoming decades [16]. In order to mitigate these risks, AI systems could be employed to ght against\nmisleading content and automated spam/phishing.\nAcknowledgments\nThe authors would like to thank the reviewers for their thoughtful and constructive feedback on this\npaper, as

well as HuggingFace for their help in open-sourcing code to run RAG models. The authors\nwould also like to thank Kyunghyun Cho and Sewon Min for productive discussions and advice. EP\nthanks supports from the NSF Graduate Research Fellowship. PL is supported by the FAIR PhD\nprogram.\nReferences\n[1] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan\nMajumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina\nStoica, Saurabh Tiwary, and Tong Wang. MS MARCO: A Human Generated MAchine\nReading COmprehension Dataset. arXiv:1611.09268 [cs], November 2016. URL http:\n//arxiv.org/abs/1611.09268. arXiv: 1611.09268.\n[2] Petr Baudiš and Jan Šediv`y. Modeling of the question answering task in the yodaqa system. In\nInternational Conference of the Cross-Language Evaluation Forum for European Languages,\npages 222-228. Springer, 2015. URL https://link.springer.com/chapter/10.1007%\n2F978-3-319-24027-5_20 .\n[3] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic Parsing on Freebase\nfrom Question-Answer Pairs. In Proceedings of the 2013 Conference on Empirical Methods\nin Natural Language Processing, pages 1533-1544, Seattle, Washington, USA, October 2013.\nAssociation for Computational Linguistics. URL http://www.aclweb.org/anthology/\nD13-1160.\n[4] Bin Bi, Chenliang Li, Chen Wu, Ming Yan, and Wei Wang. Palm: Pre-training an autoencod-\ning&autoregressive language model for context-conditioned generation. ArXiv, abs/2004.07159,\n2020. URL https://arxiv.org/abs/2004.07159.\n[5] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading Wikipedia to Answer\nOpen-Domain Questions. In Proceedings of the 55th Annual Meeting of the Association for\nComputational Linguistics (Volume 1: Long Papers), pages 1870-1879, Vancouver, Canada,\nJuly 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1171. URL\nhttps://www.aclweb.org/anthology/P17-1171.\n[6] Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Illia Polosukhin, Alexandre Lacoste, and\nJonathan Berant. Coarse-to- ne question answering for long documents. In Proceedings of the\n55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),\npages 209-220, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi:\n10.18653/v1/P17-1020. URL https://www.aclweb.org/anthology/P17-1020.\n10'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 10}, page_content='[7] Christopher Clark and Matt Gardner. Simple and Effective Multi-Paragraph Reading Compre-\nhension. arXiv:1710.10723 [cs], October 2017. URL http://arxiv.org/abs/1710.10723.\narXiv: 1710.10723.\n[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of\nDeep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Con-\nference of the North American Chapter of the Association for Computational Linguistics: Human\nLanguage Technologies, Volume 1 (Long and Short Papers), pages 4171-4186, Minneapolis,\nMinnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423.\nURL https://www.aclweb.org/anthology/N19-1423.\n[9] Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. Wiz-\nard of wikipedia: Knowledge-powered conversational agents. In International Conference on\nLearning Representations, 2019. URL https://openreview.net/forum?id=r1l73iRqKm.\n[10] Matthew Dunn, Levent Sagun,

Mike Higgins, V . Ugur Guney, V olkan Cirik, and Kyunghyun\nCho. SearchQA: A New Q&A Dataset Augmented with Context from a Search Engine.\narXiv:1704.05179 [cs], April 2017. URL http://arxiv.org/abs/1704.05179. arXiv:\n1704.05179.\n[11] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In Proceed-\nings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1:\nLong Papers), pages 889–898, Melbourne, Australia, July 2018. Association for Computational\nLinguistics. doi: 10.18653/v1/P18-1082. URL https://www.aclweb.org/anthology/\nP18-1082.\n[12] Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. ELI5:\nLong form question answering. In Proceedings of the 57th Annual Meeting of the Association\nfor Computational Linguistics, pages 3558–3567, Florence, Italy, July 2019. Association for\nComputational Linguistics. doi: 10.18653/v1/P19-1346. URL https://www.aclweb.org/\nanthology/P19-1346.\n[13] Angela Fan, Claire Gardent, Chloe Braud, and Antoine Bordes. Augmenting transformers\nwith KNN-based composite memory, 2020. URL https://openreview.net/forum?id=\nH1gx1CNKPH.\n[14] Thibault Févry, Livio Baldini Soares, Nicholas FitzGerald, Eunsol Choi, and Tom Kwiatkowski.\nEntities as experts: Sparse memory access with entity supervision. ArXiv, abs/2004.07202,\n2020. URL https://arxiv.org/abs/2004.07202.\n[15] Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen\ntau Yih, and Michel Galley. A knowledge-grounded neural conversation model. In AAAI\nConference on Arti cial Intelligence, 2018. URL https://www.aaai.org/ocs/index.php/\nAAAI/AAAI18/paper/view/16710.\n[16] Katja Grace, John Salvatier, Allan Dafoe, Baobao Zhang, and Owain Evans. When will AI\nexceed human performance? evidence from AI experts. CoRR, abs/1705.08807, 2017. URL\nhttp://arxiv.org/abs/1705.08807.\n[17] Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor O.K. Li. Search engine guided neural\nmachine translation. In AAAI Conference on Arti cial Intelligence , 2018. URL https:\n//www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17282.\n[18] Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor O.K. Li. Search engine guided neural\nmachine translation. In 32nd AAAI Conference on Arti cial Intelligence, AAAI 2018 , 32nd\nAAAI Conference on Arti cial Intelligence, AAAI 2018, pages 5133–5140. AAAI press, 2018.\n32nd AAAI Conference on Arti cial Intelligence, AAAI 2018 ; Conference date: 02-02-2018\nThrough 07-02-2018.\n[19] Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. Generating sentences by\nediting prototypes. Transactions of the Association for Computational Linguistics, 6:437–450,\n2018. doi: 10.1162/tacl_a_00030. URL https://www.aclweb.org/anthology/Q18-1031.\n11'),

Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 11}, page_content='[20] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. REALM:\nRetrieval-augmented language model pre-training. ArXiv, abs/2002.08909, 2020. URL https:\n//arxiv.org/abs/2002.08909.\n[21] Tatsunori B Hashimoto, Kelvin Guu, Yonatan Oren, and Percy S Liang. A\nretrieve-and-edit framework for predicting structured outputs. In S. Bengio,\nH. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, ed-\nitors, Advances in Neural Information Processing Systems 31 , pages 10052–\n10062. Curran Associates, Inc., 2018. URL

http://papers.nips.cc/paper/\n8209-a-retrieve-and-edit-framework-for-predicting-structured-outputs.\npdf.\n[22] Nabil Hossain, Marjan Ghazvininejad, and Luke Zettlemoyer. Simple and effective retrieve-\nedit-rerank text generation. In Proceedings of the 58th Annual Meeting of the Association for\nComputational Linguistics, pages 2532–2538, Online, July 2020. Association for Computa-\ntional Linguistics. doi: 10.18653/v1/2020.acl-main.228. URL https://www.aclweb.org/\nanthology/2020.acl-main.228.\n[23] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. arXiv\npreprint arXiv:1702.08734, 2017. URL https://arxiv.org/abs/1702.08734.\n[24] Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A Large Scale\nDistantly Supervised Challenge Dataset for Reading Comprehension. In Proceedings of the\n55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),\npages 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics.\ndoi: 10.18653/v1/P17-1147. URL https://www.aclweb.org/anthology/P17-1147.\n[25] Armand Joulin and Tomas Mikolov. Inferring algorithmic patterns with stack-\naugmented recurrent nets. In Proceedings of the 28th International Conference on\nNeural Information Processing Systems - Volume 1 , NIPS'15, page 190–198, Cam-\nbridge, MA, USA, 2015. MIT Press. URL https://papers.nips.cc/paper/\n5857-inferring-algorithmic-patterns-with-stack-augmented-recurrent-nets .\n[26] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and\nWen-tau Yih. Dense passage retrieval for open-domain question answering. arXiv preprint\narXiv:2004.04906, 2020. URL https://arxiv.org/abs/2004.04906.\n[27] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generaliza-\ntion through memorization: Nearest neighbor language models. In International Conference on\nLearning Representations, 2020. URL https://openreview.net/forum?id=HklBjCEKvH.\n[28] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua\nBengio and Yann LeCun, editors, 3rd International Conference on Learning Representations,\nICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015. URL\nhttp://arxiv.org/abs/1412.6980.\n[29] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Red eld, Michael Collins, Ankur Parikh,\nChris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Ken-\nton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob\nUszkoreit, Quoc Le, and Slav Petrov. Natural Questions: a Benchmark for Ques-\ntion Answering Research. Transactions of the Association of Computational Lin-\nguistics, 2019. URL https://tomkwiat.users.x20web.corp.google.com/papers/\nnatural-questions/main-1455-kwiatkowski.pdf .\n[30] Guillaume Lample, Alexandre Sablayrolles, Marc' Aurelio Ranzato, Ludovic Denoyer, and\nHerve Jegou. Large memory layers with product keys. In H. Wallach, H. Larochelle,\nA. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, Advances in Neural In-\nformation Processing Systems 32, pages 8548–8559. Curran Associates, Inc., 2019. URL http:\n//papers.nips.cc/paper/9061-large-memory-layers-with-product-keys.pdf .\n[31] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised\nopen domain question answering. In Proceedings of the 57th Annual Meeting of the Association\n12'), Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 12}, page_content='for Computational Linguistics, pages 6086-6096, Florence, Italy, July 2019. Association for\nComputational Linguistics. doi: 10.18653/v1/P19-1612. URL https://www.aclweb.org/\nanthology/P19-1612.\n[32] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed,\nOmer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence\npre-training for natural language generation, translation, and comprehension. arXiv preprint\narXiv:1910.13461, 2019. URL https://arxiv.org/abs/1910.13461.\n[33] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A diversity-promoting\nobjective function for neural conversation models. In Proceedings of the 2016 Conference of the\nNorth American Chapter of the Association for Computational Linguistics: Human Language\nTechnologies, pages 110-119, San Diego, California, June 2016. Association for Computational\nLinguistics. doi: 10.18653/v1/N16-1014. URL https://www.aclweb.org/anthology/\nN16-1014.\n[34] Margaret Li, Jason Weston, and Stephen Roller. Acute-eval: Improved dialogue evaluation\nwith optimized questions and multi-turn comparisons. ArXiv, abs/1909.03087, 2019. URL\nhttps://arxiv.org/abs/1909.03087.\n[35] Hairong Liu, Mingbo Ma, Liang Huang, Hao Xiong, and Zhongjun He. Robust neural machine\ntranslation with joint textual and phonetic embedding. In Proceedings of the 57th Annual\nMeeting of the Association for Computational Linguistics, pages 3044-3049, Florence, Italy,\nJuly 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1291. URL\nhttps://www.aclweb.org/anthology/P19-1291.\n[36] Peter J. Liu*, Mohammad Saleh*, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser,\nand Noam Shazeer. Generating wikipedia by summarizing long sequences. In International\nConference on Learning Representations, 2018. URL https://openreview.net/forum?\nid=Hyg0vbWC-.\n[37] Yury A. Malkov and D. A. Yashunin. Ef cient and robust approximate nearest neighbor search\nusing hierarchical navigable small world graphs. IEEE Transactions on Pattern Analysis and\nMachine Intelligence, 42:824-836, 2016. URL https://arxiv.org/abs/1603.09320.\n[38] Gary Marcus. The next decade in ai: four steps towards robust arti cial intelligence. arXiv\npreprint arXiv:2002.06177, 2020. URL https://arxiv.org/abs/2002.06177.\n[39] Luca Massarelli, Fabio Petroni, Aleksandra Piktus, Myle Ott, Tim Rocktäschel, Vassilis\nPlachouras, Fabrizio Silvestri, and Sebastian Riedel. How decoding strategies affect the\nveri ability of generated text. arXiv preprint arXiv:1911.03587 , 2019. URL https:\n//arxiv.org/abs/1911.03587.\n[40] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia,\nBoris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed\nprecision training. In ICLR, 2018. URL https://openreview.net/forum?id=r1gs9JgRZ.\n[41] Nikita Moghe, Siddhartha Arora, Suman Banerjee, and Mitesh M. Khapra. Towards exploit-\ning background knowledge for building conversation systems. In Proceedings of the 2018\nConference on Empirical Methods in Natural Language Processing, pages 2322-2332, Brus-\nsels, Belgium, October-November 2018. Association for Computational Linguistics. doi:\n10.18653/v1/D18-1255. URL https://www.aclweb.org/anthology/D18-1255.\n[42] Preksha Nema and Mitesh M.

Khapra. Towards a better metric for evaluating question generation\nsystems. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language\nProcessing, pages 3950-3959, Brussels, Belgium, October-November 2018. Association for\nComputational Linguistics. doi: 10.18653/v1/D18-1429. URL https://www.aclweb.org/\nanthology/D18-1429.\n[43] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder,\nand Li Deng. MS MARCO: A human generated machine reading comprehension dataset. In\nTarek Richard Besold, Antoine Bordes, Artur S. d'Avila Garcez, and Greg Wayne, editors,\nProceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic\n13'),
Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 13}, page_content='approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing\nSystems (NIPS 2016), Barcelona, Spain, December 9, 2016, volume 1773 of CEUR Workshop\nProceedings. CEUR-WS.org, 2016. URL http://ceur-ws.org/Vol-1773/CoCoNIPS_\n2016_paper9.pdf.\n[44] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with BERT. arXiv preprint\narXiv:1901.04085, 2019. URL https://arxiv.org/abs/1901.04085.\n[45] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier,\nand Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In Proceedings\nof the 2019 Conference of the North American Chapter of the Association for Computational\nLinguistics (Demonstrations), pages 48-53, Minneapolis, Minnesota, June 2019. Association\nfor Computational Linguistics. doi: 10.18653/v1/N19-4009. URL https://www.aclweb.\norg/anthology/N19-4009.\n[46] Ethan Perez, Siddharth Karamcheti, Rob Fergus, Jason Weston, Douwe Kiela, and Kyunghyun\nCho. Finding generalizable evidence by learning to convince q&a models. In Proceedings\nof the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th\nInternational Joint Conference on Natural Language Processing (EMNLP-IJCNLP) , pages\n2402-2411, Hong Kong, China, November 2019. Association for Computational Linguistics.\ndoi: 10.18653/v1/D19-1244. URL https://www.aclweb.org/anthology/D19-1244.\n[47] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu,\nand Alexander Miller. Language models as knowledge bases? In Proceedings of the 2019\nConference on Empirical Methods in Natural Language Processing and the 9th International\nJoint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 2463-2473, Hong\nKong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/\nD19-1250. URL https://www.aclweb.org/anthology/D19-1250.\n[48] Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H.\nMiller, and Sebastian Riedel. How context affects language models' factual predictions. In\nAutomated Knowledge Base Construction, 2020. URL https://openreview.net/forum?\nid=025X0zPfn.\n[49] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Im-\nproving Language Understanding by Generative Pre-Training, 2018. URL\nhttps://s3-us-west-2.amazonaws.com/openai-assets/research-covers/\nlanguage-unsupervised/language_understanding_paper.pdf.\n[50] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya\nSutskever. Language models are

unsupervised multitask learners, 2019.
URL\nhttps://d4mucfpksywv.cloudfront.net/better-language-models/language_\nmodels_are_unsupervised_multitask_learners.pdf.\n[51] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena,\nYanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a uni ed\ntext-to-text transformer. arXiv e-prints, 2019. URL https://arxiv.org/abs/1910.10683.\n[52] Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into\nthe parameters of a language model? arXiv e-prints, 2020. URL https://arxiv.org/abs/\n2002.08910.\n[53] Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and\nbeyond. Found. Trends Inf. Retr., 3(4):333-389, April 2009. ISSN 1554-0669. doi: 10.1561/\n1500000019. URL https://doi.org/10.1561/1500000019.\n[54] Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-V oss, Jeff Wu, Alec\nRadford, and Jian-Bing Wang. Release strategies and the social impacts of language models.\nArXiv, abs/1908.09203, 2019.\n[55] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory net-\nworks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors,Advances\nin Neural Information Processing Systems 28, pages 2440-2448. Curran Associates, Inc., 2015.\nURL http://papers.nips.cc/paper/5846-end-to-end-memory-networks.pdf .\n14'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 14}, page_content='[56] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a\nlarge-scale dataset for fact extraction and VERi cation. In Proceedings of the 2018 Conference\nof the North American Chapter of the Association for Computational Linguistics: Human\nLanguage Technologies, Volume 1 (Long Papers), pages 809-819, New Orleans, Louisiana,\nJune 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1074. URL\nhttps://www.aclweb.org/anthology/N18-1074.\n[57] James H. Thorne and Andreas Vlachos. Avoiding catastrophic forgetting in mitigating model\nbiases in sentence-pair classi cation with elastic weight consolidation. ArXiv, abs/2004.14366,\n2020. URL https://arxiv.org/abs/2004.14366.\n[58] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez,\nŁ ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V . Luxburg,\nS. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors,Advances in Neural\nInformation Processing Systems 30, pages 5998-6008. Curran Associates, Inc., 2017. URL\nhttp://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf .\n[59] Ashwin Vijayakumar, Michael Cogswell, Ramprasaath Selvaraju, Qing Sun, Stefan Lee, David\nCrandall, and Dhruv Batra. Diverse beam search for improved description of complex scenes.\nAAAI Conference on Arti cial Intelligence, 2018. URL https://www.aaai.org/ocs/index.\nphp/AAAI/AAAI18/paper/view/17329.\n[60] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman.\nGLUE: A multi-task benchmark and analysis platform for natural language understanding.\nIn Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting\nNeural Networks for NLP, pages 353-355, Brussels, Belgium, November 2018. Association for\nComputational Linguistics. doi: 10.18653/v1/W18-5446. URL https://www.aclweb.org/\nanthology/W18-5446.\n[61]

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix\nHill, Omer Levy, and Samuel Bowman. SuperGLUE: A Stickier Benchmark for General-\nPurpose Language Understanding Systems. In H. Wallach, H. Larochelle, A. Beygelzimer,\nF. d\\textquotesingle Alché-Buc, E. Fox, and R. Garnett, editors,Advances in Neural Information\nProcessing Systems 32, pages 3261–3275. Curran Associates, Inc., 2019. URL https://\narxiv.org/abs/1905.00537.\n[62] Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang,\nGerry Tesauro, Bowen Zhou, and Jing Jiang. R3: Reinforced ranker-reader for open-domain\nquestion answering. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, Proceedings of\nthe Thirty-Second AAAI Conference on Arti cial Intelligence, (AAAI-18), the 30th innovative\nApplications of Arti cial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational\nAdvances in Arti cial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7,\n2018, pages 5981–5988. AAAI Press, 2018. URL https://www.aaai.org/ocs/index.\nphp/AAAI/AAAI18/paper/view/16712.\n[63] Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang,\nTim Klinger, Gerald Tesauro, and Murray Campbell. Evidence aggregation for answer re-\nranking in open-domain question answering. In ICLR, 2018. URL https://openreview.\nnet/forum?id=rJl3yM-Ab.\n[64] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. In Yoshua Bengio\nand Yann LeCun, editors, 3rd International Conference on Learning Representations, ICLR\n2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings , 2015. URL\nhttp://arxiv.org/abs/1410.3916.\n[65] Jason Weston, Emily Dinan, and Alexander Miller. Retrieve and re ne: Improved sequence\ngeneration models for dialogue. In Proceedings of the 2018 EMNLP Workshop SCAI: The 2nd\nInternational Workshop on Search-Oriented Conversational AI, pages 87–92, Brussels, Belgium,\nOctober 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5713. URL\nhttps://www.aclweb.org/anthology/W18-5713.\n15'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 15}, page_content='[66] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony\nMoi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer,\nPatrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain\nGugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers:\nState-of-the-art natural language processing. ArXiv, abs/1910.03771, 2019.\n[67] Shiyue Zhang and Mohit Bansal. Addressing semantic drift in question generation for semi-\nsupervised question answering. In Proceedings of the 2019 Conference on Empirical Meth-\nods in Natural Language Processing and the 9th International Joint Conference on Natural\nLanguage Processing (EMNLP-IJCNLP) , pages 2495–2509, Hong Kong, China, Novem-\nber 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1253. URL\nhttps://www.aclweb.org/anthology/D19-1253.\n[68] Wanjun Zhong, Jingjing Xu, Duyu Tang, Zenan Xu, Nan Duan, Ming Zhou, Jiahai Wang, and\nJian Yin. Reasoning over semantic-level graph for fact checking. ArXiv, abs/1909.03745, 2019.\nURL https://arxiv.org/abs/1909.03745.\n16'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 16}, page_content='Appendices for Retrieval-Augmented

Generation for\nKnowledge-Intensive NLP Tasks\nA Implementation Details\nFor Open-domain QA we report test numbers using 15 retrieved documents for RAG-Token models.\nFor RAG-Sequence models, we report test results using 50 retrieved documents, and we use the\nThorough Decoding approach since answers are generally short. We use greedy decoding for QA as\nwe did not nd beam search improved results. For Open-MSMarco and Jeopardy question generation,\nwe report test numbers using ten retrieved documents for both RAG-Token and RAG-Sequence,\nand we also train a BART-large model as a baseline. We use a beam size of four, and use the Fast\nDecoding approach for RAG-Sequence models, as Thorough Decoding did not improve performance.\nB Human Evaluation\nFigure 4: Annotation interface for human evaluation of factuality. A pop-out for detailed instructions\nand a worked example appear when clicking "view tool guide".\nFigure 4 shows the user interface for human evaluation. To avoid any biases for screen position,\nwhich model corresponded to sentence A and sentence B was randomly selected for each example.\nAnnotators were encouraged to research the topic using the internet, and were given detailed instruc-\ntions and worked examples in a full instructions tab. We included some gold sentences in order to\nassess the accuracy of the annotators. Two annotators did not perform well on these examples and\ntheir annotations were removed from the results.\nC Training setup Details\nWe train all RAG models and BART baselines using Fairseq [45].2 We train with mixed precision\n oating point arithmetic [40], distributing training across 8, 32GB NVIDIA V100 GPUs, though\ntraining and inference can be run on one GPU. We nd that doing Maximum Inner Product Search\nwith FAISS is suf ciently fast on CPU, so we store document index vectors on CPU, requiring 100\nGB of CPU memory for all of Wikipedia. After submission, We have ported our code to HuggingFace\nTransformers [66]3, which achieves equivalent performance to the previous version but is a cleaner\nand easier to use implementation. This version is also open-sourced. We also compress the document\nindex using FAISS's compression tools, reducing the CPU memory requirement to 36GB. Scripts to\nrun experiments with RAG can be found at https://github.com/huggingface/transformers/\nblob/master/examples/rag/README.md and an interactive demo of a RAG model can be found\nat https://huggingface.co/rag/\n2https://github.com/pytorch/fairseq\n3https://github.com/huggingface/transformers\n17Viewfull instructions\nWhich sentence is more factually true?\nView tool guide\nSelect an option\nSubject\n: Hemingway\nNote:Somequestionsare\nSentence A is more\n1\ncontrol questions.Werequire\ntrue\nSentence A: "The Sun Also Rises" is a novel by this author of "A\ngood accuracy on our control\nFarewell to Arms"\nSentenceBismore\n2\nquestions to accept\ntrue\nresponses.\nSentenceB:Thisauthorof"TheSunAlsoRises"wasbornin\nBoth sentences are\n3\nHavana, Cuba, the son of Spanish immigrants\ntrue\nIndicatewhichoneofthe\nfollowing sentences is more\nBoth sentences are\ncompletely untrue\nfactuallytruewithrespectto\nthe subject. Using the\ninternet to check whether\nthesentencesaretrueis\nencouraged.'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 17}, page_content='D Further Details on Open-Domain QA\nFor open-domain QA, multiple answer annotations are often available for a given

question. These\nanswer annotations are exploited by extractive models during training as typically all the answer\nannotations are used to nd matches within documents when preparing training data. For RAG, we\nalso make use of multiple annotation examples for Natural Questions and WebQuestions by training\nthe model with each (q,a) pair separately, leading to a small increase in accuracy. For TriviaQA,\nthere are often many valid answers to a given question, some of which are not suitable training targets,\nsuch as emoji or spelling variants. For TriviaQA, we lter out answer candidates if they do not occur\nin top 1000 documents for the query.\nCuratedTrec preprocessing The answers for CuratedTrec are given in the form of regular expres-\nsions, which has been suggested as a reason why it is unsuitable for answer-generation models [20].\nTo overcome this, we use a pre-processing step where we rst retrieve the top 1000 documents for\neach query, and use the answer that most frequently matches the regex pattern as the supervision\ntarget. If no matches are found, we resort to a simple heuristic: generate all possible permutations for\neach regex, replacing non-deterministic symbols in the regex nested tree structure with a whitespace.\nTriviaQA Evaluation setups The open-domain QA community customarily uses public develop-\nment datasets as test datasets, as test data for QA datasets is often restricted and dedicated to reading\ncompehension purposes. We report our results using the datasets splits used in DPR [26], which are\nconsistent with common practice in Open-domain QA. For TriviaQA, this test dataset is the public\nTriviaQA Web Development split. Roberts et al.[52] used the TriviaQA of cial Wikipedia test set\ninstead. Févry et al. [14] follow this convention in order to compare with Roberts et al. [52] (See\nappendix of [14]). We report results on both test sets to enable fair comparison to both approaches.\nWe nd that our performance is much higher using the of cial Wiki test set, rather than the more\nconventional open-domain test set, which we attribute to the of cial Wiki test set questions being\nsimpler to answer from Wikipedia.\nE Further Details on FEVER\nFor FEVER classi cation, we follow the practice from [32], and rst re-generate the claim, and\nthen classify using the representation of the nal hidden state, before nally marginalizing across\ndocuments to obtain the class probabilities. The FEVER task traditionally has two sub-tasks. The\n rst is to classify the claim as either "Supported", "Refuted" or "Not Enough Info", which is the task\nwe explore in the main paper. FEVER's other sub-task involves extracting sentences from Wikipedia\nas evidence supporting the classi cation prediction. As FEVER uses a different Wikipedia dump to\nus, directly tackling this task is not straightforward. We hope to address this in future work.\nF Null Document Probabilities\nWe experimented with adding "Null document" mechanism to RAG, similar to REALM [20] in order\nto model cases where no useful information could be retrieved for a given input. Here, ifkdocuments\nwere retrieved, we would additionally "retrieve" an empty document and predict a logit for the null\ndocument, before marginalizing over k+ 1predictions. We explored modelling this null document\nlogit by learning (i) a document embedding for the null document, (ii) a static learnt bias term, or\n(iii) a neural network to predict the logit. We did not nd that these improved performance, so in\nthe interests of simplicity, we omit them. For Open MS-MARCO, where useful retrieved

```
documents\ncannot always be retrieved, we observe that the model learns to
always retrieve a particular set of\ndocuments for questions that are less
likely to bene t from retrieval, suggesting that null document\nmechanisms may
not be necessary for RAG.\nG Parameters\nOur RAG models contain the trainable
parameters for the BERT-base query and document encoder of\nDPR, with 110M
parameters each (although we do not train the document encoder ourselves)
and\n406M trainable parameters from BART-large, 406M parameters, making a total
of 626M trainable\n18'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 18}, page_content='Table 7: Number of instances in the
datasets used. *A hidden subset of this data is used for evaluation\nTask Train
Development Test\nNatural Questions 79169 8758 3611\nTriviaQA 78786 8838
11314\nWebQuestions 3418 362 2033\nCuratedTrec 635 134 635\nJeopardy Question
Generation 97392 13714 26849\nMS-MARCO 153726 12468 101093*\nFEVER-3-way 145450
10000 10000\nFEVER-2-way 96966 6666 6666\nparameters. The best performing
"closed-book" (parametric only) open-domain QA model is T5-11B\nwith 11 Billion
trainable parameters. The T5 model with the closest number of parameters to
our\nmodels is T5-large (770M parameters), which achieves a score of 28.9 EM on
Natural Questions [52],\nsubstantially below the 44.5 that RAG-Sequence
achieves, indicating that hybrid parametric/non-\nparametric models require far
fewer trainable parameters for strong open-domain QA performance.\nThe non-
parametric memory index does not consist of trainable parameters, but does
consists of 21M\n728 dimensional vectors, consisting of 15.3B values. These can
be easily be stored at 8-bit  oating\npoint precision to manage memory and disk
footprints.\nH Retrieval Collapse\nIn preliminary experiments, we observed that
for some tasks such as story generation [ 11], the\nretrieval component would
"collapse" and learn to retrieve the same documents regardless of the\ninput. In
these cases, once retrieval had collapsed, the generator would learn to ignore
the documents,\nand the RAG model would perform equivalently to BART. The
collapse could be due to a less-explicit\nrequirement for factual knowledge in
some tasks, or the longer target sequences, which could result\nin less
informative gradients for the retriever. Perez et al.[46] also found spurious
retrieval results\nwhen optimizing a retrieval component in order to improve
performance on downstream tasks.\nI Number of instances per dataset\nThe number
of training, development and test datapoints in each of our datasets is shown in
Table 7.\n19')]
```

[22]:
```
!pip install sentence_transformers
```

```
Requirement already satisfied: sentence_transformers in
/usr/local/lib/python3.10/dist-packages (3.2.1)
Requirement already satisfied: transformers<5.0.0,>=4.41.0 in
/usr/local/lib/python3.10/dist-packages (from sentence_transformers) (4.44.2)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages
(from sentence_transformers) (4.66.5)
Requirement already satisfied: torch>=1.11.0 in /usr/local/lib/python3.10/dist-
packages (from sentence_transformers) (2.5.0+cu121)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-
```

```
packages (from sentence_transformers) (1.5.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages
(from sentence_transformers) (1.13.1)
Requirement already satisfied: huggingface-hub>=0.20.0 in
/usr/local/lib/python3.10/dist-packages (from sentence_transformers) (0.24.7)
Requirement already satisfied: Pillow in /usr/local/lib/python3.10/dist-packages
(from sentence_transformers) (10.4.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from huggingface-hub>=0.20.0->sentence_transformers) (3.16.1)
Requirement already satisfied: fsspec>=2023.5.0 in
/usr/local/lib/python3.10/dist-packages (from huggingface-
hub>=0.20.0->sentence_transformers) (2024.6.1)
Requirement already satisfied: packaging>=20.9 in
/usr/local/lib/python3.10/dist-packages (from huggingface-
hub>=0.20.0->sentence_transformers) (24.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-
packages (from huggingface-hub>=0.20.0->sentence_transformers) (6.0.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from huggingface-hub>=0.20.0->sentence_transformers) (2.32.3)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.10/dist-packages (from huggingface-
hub>=0.20.0->sentence_transformers) (4.12.2)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-
packages (from torch>=1.11.0->sentence_transformers) (3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages
(from torch>=1.11.0->sentence_transformers) (3.1.4)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.10/dist-
packages (from torch>=1.11.0->sentence_transformers) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from
sympy==1.13.1->torch>=1.11.0->sentence_transformers) (1.3.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-
packages (from transformers<5.0.0,>=4.41.0->sentence_transformers) (1.26.4)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.10/dist-packages (from
transformers<5.0.0,>=4.41.0->sentence_transformers) (2024.9.11)
Requirement already satisfied: safetensors>=0.4.1 in
/usr/local/lib/python3.10/dist-packages (from
transformers<5.0.0,>=4.41.0->sentence_transformers) (0.4.5)
Requirement already satisfied: tokenizers<0.20,>=0.19 in
/usr/local/lib/python3.10/dist-packages (from
transformers<5.0.0,>=4.41.0->sentence_transformers) (0.19.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn->sentence_transformers) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-
learn->sentence_transformers) (3.5.0)
Requirement already satisfied: MarkupSafe>=2.0 in
```

```
/usr/local/lib/python3.10/dist-packages (from
jinja2->torch>=1.11.0->sentence_transformers) (3.0.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->huggingface-
hub>=0.20.0->sentence_transformers) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests->huggingface-hub>=0.20.0->sentence_transformers) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->huggingface-
hub>=0.20.0->sentence_transformers) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests->huggingface-
hub>=0.20.0->sentence_transformers) (2024.8.30)
```

[23]:
```
from langchain.embeddings import HuggingFaceEmbeddings
embeddings_model_name = 'sentence-transformers/all-mpnet-base-v2'
embeddings = HuggingFaceEmbeddings(model_name="sentence-transformers/
 ↪all-mpnet-base-v2")
```

[24]:
```
from langchain.text_splitter import RecursiveCharacterTextSplitter
text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000,␣
 ↪chunk_overlap=20)
docs = text_splitter.split_documents(pages)
```

[25]:
```
docs
```

[25]:
```
[Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 0}, page_content='Retrieval-Augmented Generation
for\nKnowledge-Intensive NLP Tasks\nPatrick Lewis†‡, Ethan Perez ,\nAleksandra
Piktus†, Fabio Petroni†, Vladimir Karpukhin†, Naman Goyal†, Heinrich
Küttler†,\nMike Lewis†, Wen-tau Yih†, Tim Rocktäschel†‡, Sebastian Riedel†‡,
Douwe Kiela†\n†Facebook AI Research; ‡University College London;  New York
University;\nplewis@fb.com\nAbstract\nLarge pre-trained language models have
been shown to store factual knowledge\nin their parameters, and achieve state-
of-the-art results when  ne-tuned on down-\nstream NLP tasks. However, their
ability to access and precisely manipulate knowl-\nedge is still limited, and
hence on knowledge-intensive tasks, their performance\nlags behind task-speci c
architectures. Additionally, providing provenance for their\ndecisions and
updating their world knowledge remain open research problems. Pre-\ntrained
models with a differentiable access mechanism to explicit non-parametric'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 0}, page_content='memory have so far been only investigated
for extractive downstream tasks. We\nexplore a general-purpose  ne-tuning recipe
for retrieval-augmented generation\n(RAG) - models which combine pre-trained
parametric and non-parametric mem-\nory for language generation. We introduce
RAG models where the parametric\nmemory is a pre-trained seq2seq model and the
non-parametric memory is a dense\nvector index of Wikipedia, accessed with a
```

pre-trained neural retriever. We com-\npare two RAG formulations, one which
conditions on the same retrieved passages\nacross the whole generated sequence,
and another which can use different passages\nper token. We ne-tune and
evaluate our models on a wide range of knowledge-\nintensive NLP tasks and set
the state of the art on three open domain QA tasks,\noutperforming parametric
seq2seq models and task-speci c retrieve-and-extract\narchitectures. For
language generation tasks, we nd that RAG models generate'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 0}, page_content='more speci c, diverse and factual language
than a state-of-the-art parametric-only\nseq2seq baseline.\n1 Introduction\nPre-
trained neural language models have been shown to learn a substantial amount of
in-depth knowl-\nedge from data [47]. They can do so without any access to an
external memory, as a parameterized\nimplicit knowledge base [51, 52]. While
this development is exciting, such models do have down-\nsides: They cannot
easily expand or revise their memory, can’t straightforwardly provide insight
into\ntheir predictions, and may produce "hallucinations" [38]. Hybrid models
that combine parametric\nmemory with non-parametric (i.e., retrieval-based)
memories [20, 26, 48] can address some of these\nissues because knowledge can be
directly revised and expanded, and accessed knowledge can be\ninspected and
interpreted. REALM [20] and ORQA [31], two recently introduced models
that\ncombine masked language models [8] with a differentiable retriever, have
shown promising results,'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 0}, page_content='arXiv:2005.11401v4 [cs.CL] 12 Apr
2021'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 1}, page_content='The\tDivine\nComedy\t(x) q \nQuery
\nEncoder \nq(x) \nMIPS p \nGenerator\xa0p \n(Parametric) \nMargin- \nalize \n
This\t14th\tcentury\twork\nis\tdivided\tinto\t3\nsections:\t"Inferno",\n"Purgato
rio"\t&\n"Paradiso"\t\t\t\t\t\t\t\t\t\t(y)\nEnd-to-End Backprop through q
and\xa0p \nBarack\tObama\twas\nborn\tin\tHawaii.(x)\nFact Veri cation: Fact
Query\nsupports\t(y)\nQuestion Generation\nFact Veri cation:\nLabel
Generation\nDocument \nIndex \nDefine\t"middle\tear"(x)\nQuestion
Answering:\nQuestion Query\nThe\tmiddle\tear\tincludes\nthe\ttympanic\tcavity\ta
nd\nthe\tthree\tossicles.\t\t(y)\nQuestion Answering:\nAnswer
GenerationRetriever p \n(Non-Parametric) \nz 4 \nz 3 \nz 2 \nz 1 \nd(z)
\nJeopardy Question\nGeneration:\nAnswer Query\nFigure 1: Overview of our
approach. We combine a pre-trained retriever (Query Encoder + Document\nIndex)
with a pre-trained seq2seq model (Generator) and ne-tune end-to-end. For query
x , we use\nMaximum Inner Product Search (MIPS) to nd the top-K documents z i .
For nal prediction y , we'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 1}, page_content='treat z as a latent variable and
marginalize over seq2seq predictions given different documents.\nbut have only
explored open-domain extractive question answering. Here, we bring hybrid
parametric\nand non-parametric memory to the "workhorse of NLP," i.e. sequence-
to-sequence (seq2seq) models.\nWe endow pre-trained, parametric-memory

generation models with a non-parametric memory through\na general-purpose ne-
tuning approach which we refer to as retrieval-augmented generation (RAG).\nWe
build RAG models where the parametric memory is a pre-trained seq2seq
transformer, and the\nnon-parametric memory is a dense vector index of
Wikipedia, accessed with a pre-trained neural\nretriever. We combine these
components in a probabilistic model trained end-to-end (Fig. 1). The\nretriever
(Dense Passage Retriever [26], henceforth DPR) provides latent documents
conditioned on\nthe input, and the seq2seq model (BART [32]) then conditions on
these latent documents together with'),
Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 1}, page_content='the input to generate the output. We
marginalize the latent documents with a top-K approximation,\neither on a per-
output basis (assuming the same document is responsible for all tokens) or a
per-token\nbasis (where different documents are responsible for different
tokens). Like T5 [51] or BART, RAG\ncan be ne-tuned on any seq2seq task,
whereby both the generator and retriever are jointly learned.\nThere has been
extensive previous work proposing architectures to enrich systems with non-
parametric\nmemory which are trained from scratch for speci c tasks, e.g. memory
networks [64, 55], stack-\naugmented networks [25] and memory layers [30]. In
contrast, we explore a setting where both\nparametric and non-parametric memory
components are pre-trained and pre-loaded with extensive\nknowledge. Crucially,
by using pre-trained access mechanisms, the ability to access knowledge
is\npresent without additional training.'),
Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 1}, page_content='Our results highlight the bene ts of
combining parametric and non-parametric memory with genera-\ntion for knowledge-
intensive tasks-tasks that humans could not reasonably be expected to
perform\nwithout access to an external knowledge source. Our RAG models achieve
state-of-the-art results\non open Natural Questions [29], WebQuestions [3] and
CuratedTrec [2] and strongly outperform\nrecent approaches that use specialised
pre-training objectives on TriviaQA [24]. Despite these being\nextractive tasks,
we nd that unconstrained generation outperforms previous extractive
approaches.\nFor knowledge-intensive generation, we experiment with MS-MARCO [1]
and Jeopardy question\ngeneration, and we nd that our models generate responses
that are more factual, speci c, and\ndiverse than a BART baseline. For FEVER
[56] fact veri cation, we achieve results within 4.3% of\nstate-of-the-art
pipeline models which use strong retrieval supervision. Finally, we demonstrate
that'),
Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 1}, page_content='the non-parametric memory can be replaced
to update the models' knowledge as the world changes.1\n2 Methods\nWe explore
RAG models, which use the input sequencex to retrieve text documents z and use
them\nas additional context when generating the target sequence y . As shown in
Figure 1, our models\nleverage two components: (i) a retriever p (z |x) with
parameters   that returns (top-K truncated)\ndistributions over text passages
given a query x and (ii) a generator p (y i |x,z,y 1:i −1) parametrized\n1Code
to run experiments with RAG has been open-sourced as part of the HuggingFace

```
Transform-\ners Library [66] and can be found at
https://github.com/huggingface/transformers/blob/master/\nexamples/rag/. An
interactive demo of RAG models can be found at https://huggingface.co/rag/\n2'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 2}, page_content='by  that generates a current token based
on a context of the previous i-1 tokens y1:i-1, the original\ninput xand a
retrieved passage z.\nTo train the retriever and generator end-to-end, we treat
the retrieved document as a latent variable.\nWe propose two models that
marginalize over the latent documents in different ways to produce
a\ndistribution over generated text. In one approach, RAG-Sequence, the model
uses the same document\nto predict each target token. The second approach, RAG-
Token, can predict each target token based\non a different document. In the
following, we formally introduce both models and then describe the\np  and p
components, as well as the training and decoding procedure.\n2.1 Models\nRAG-
Sequence Model The RAG-Sequence model uses the same retrieved document to
generate\nthe complete sequence. Technically, it treats the retrieved document
as a single latent variable that\nis marginalized to get the seq2seq probability
p(y|x) via a top-K approximation. Concretely, the'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 2}, page_content='top K documents are retrieved using the
retriever, and the generator produces the output sequence\nprobability for each
document, which are then marginalized,\npRAG-Sequence(y|x)
 \n \nz top-k(p(·|x))\np (z|x)p (y|x,z)
=\n \nz top-k(p(·|x))\np (z|x)\nN \ni\np (yi|x,z,y 1:i-1)\nRAG-Token Model In
the RAG-Token model we can draw a different latent document for each\ntarget
token and marginalize accordingly. This allows the generator to choose content
from several\ndocuments when producing an answer. Concretely, the top K
documents are retrieved using the\nretriever, and then the generator produces a
distribution for the next output token for each document,\nbefore marginalizing,
and repeating the process with the following output token, Formally, we
de ne:\npRAG-Token(y|x)  \nN \ni\n \nz top-k(p(·|x))\np (z|x)p (yi|x,z,y
1:i-1)\nFinally, we note that RAG can be used for sequence classi cation tasks
by considering the target class'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 2}, page_content='as a target sequence of length one, in
which case RAG-Sequence and RAG-Token are equivalent.\n2.2 Retriever: DPR\nThe
retrieval component p (z|x) is based on DPR [26]. DPR follows a bi-encoder
architecture:\np (z|x)  exp\n(\nd(z) q(x)\n)\nd(z) =BERTd(z), q(x)
=BERTq(x)\nwhere d(z) is a dense representation of a document produced by a
BERTBASE document encoder [8],\nand q(x) a query representation produced by a
query encoder, also based on BERTBASE. Calculating\ntop-k(p (·|x)), the list of
kdocuments zwith highest prior probability p (z|x), is a Maximum Inner\nProduct
Search (MIPS) problem, which can be approximately solved in sub-linear time
[23]. We use\na pre-trained bi-encoder from DPR to initialize our retriever and
to build the document index. This\nretriever was trained to retrieve documents
which contain answers to TriviaQA [24] questions and\nNatural Questions [29]. We
refer to the document index as the non-parametric memory.\n2.3 Generator:
```

BART'),
Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 2}, page_content='2.3 Generator: BART\nThe generator component p (yi|x,z,y 1:i-1) could be modelled using any encoder-decoder. We use\nBART-large [32], a pre-trained seq2seq transformer [58] with 400M parameters. To combine the input\nxwith the retrieved content zwhen generating from BART, we simply concatenate them. BART was\npre-trained using a denoising objective and a variety of different noising functions. It has obtained\nstate-of-the-art results on a diverse set of generation tasks and outperforms comparably-sized T5\nmodels [32]. We refer to the BART generator parameters as the parametric memory henceforth.\n2.4 Training\nWe jointly train the retriever and generator components without any direct supervision on what\ndocument should be retrieved. Given a ne-tuning training corpus of input/output pairs (xj,yj), we\n3'),
Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 3}, page_content='minimize the negative marginal log-likelihood of each target, \nj-log p(yj|xj) using stochastic\ngradient descent with Adam [28]. Updating the document encoder BERTd during training is costly as\nit requires the document index to be periodically updated as REALM does during pre-training [20].\nWe do not nd this step necessary for strong performance, and keep the document encoder (and\nindex) xed, only ne-tuning the query encoder BERTq and the BART generator.\n2.5 Decoding\nAt test time, RAG-Sequence and RAG-Token require different ways to approximatearg maxyp(y|x).\nRAG-Token The RAG-Token model can be seen as a standard, autoregressive seq2seq genera-\ntor with transition probability: p \n (yi|x,y1:i-1) = \nz top-k(p(·|x)) p (zi|x)p (yi|x,zi,y1:i-1) To\ndecode, we can plug p \n (yi|x,y1:i-1) into a standard beam decoder.\nRAG-Sequence For RAG-Sequence, the likelihood p(y|x) does not break into a conventional per-'),
Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 3}, page_content='token likelihood, hence we cannot solve it with a single beam search. Instead, we run beam search for\neach document z, scoring each hypothesis using p (yi|x,z,y 1:i-1). This yields a set of hypotheses\nY, some of which may not have appeared in the beams of all documents. To estimate the probability\nof an hypothesis y we run an additional forward pass for each document z for which y does not\nappear in the beam, multiply generator probability with p (z|x) and then sum the probabilities across\nbeams for the marginals. We refer to this decoding procedure as "Thorough Decoding." For longer\noutput sequences, |Y|can become large, requiring many forward passes. For more ef cient decoding,\nwe can make a further approximation that p (y|x,zi) 0 where ywas not generated during beam\nsearch from x,zi. This avoids the need to run additional forward passes once the candidate set Y has\nbeen generated. We refer to this decoding procedure as "Fast Decoding."\n3 Experiments'),
Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 3}, page_content='3 Experiments\nWe experiment with RAG in a wide range of knowledge-intensive tasks. For all experiments, we use\na single Wikipedia dump for our non-parametric knowledge source. Following Lee et al.

[31] and\nKarpukhin et al. [26], we use the December 2018 dump. Each Wikipedia article is split into disjoint\n100-word chunks, to make a total of 21M documents. We use the document encoder to compute an\nembedding for each document, and build a single MIPS index using FAISS [23] with a Hierarchical\nNavigable Small World approximation for fast retrieval [37]. During training, we retrieve the top\nkdocuments for each query. We consider k {5,10}for training and set kfor test time using dev\ndata. We now discuss experimental details for each task.\n3.1 Open-domain Question Answering\nOpen-domain question answering (QA) is an important real-world application and common testbed\nfor knowledge-intensive tasks [20]. We treat questions and answers as input-output text pairs (x,y)'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 3}, page_content='and train RAG by directly minimizing the negative log-likelihood of answers. We compare RAG to\nthe popular extractive QA paradigm [5, 7, 31, 26], where answers are extracted spans from retrieved\ndocuments, relying primarily on non-parametric knowledge. We also compare to "Closed-Book\nQA" approaches [52], which, like RAG, generate answers, but which do not exploit retrieval, instead\nrelying purely on parametric knowledge. We consider four popular open-domain QA datasets: Natural\nQuestions (NQ) [29], TriviaQA (TQA) [24]. WebQuestions (WQ) [3] and CuratedTrec (CT) [2]. As\nCT and WQ are small, we follow DPR [26] by initializing CT and WQ models with our NQ RAG\nmodel. We use the same train/dev/test splits as prior work [31, 26] and report Exact Match (EM)\nscores. For TQA, to compare with T5 [52], we also evaluate on the TQA Wiki test set.\n3.2 Abstractive Question Answering\nRAG models can go beyond simple extractive QA and answer questions with free-form, abstractive'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 3}, page_content='text generation. To test RAG’s natural language generation (NLG) in a knowledge-intensive setting,\nwe use the MSMARCO NLG task v2.1 [43]. The task consists of questions, ten gold passages\nretrieved from a search engine for each question, and a full sentence answer annotated from the\nretrieved passages. We do not use the supplied passages, only the questions and answers, to treat\n4'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 4}, page_content='MSMARCO as an open-domain abstractive QA task. MSMARCO has some questions that cannot be\nanswered in a way that matches the reference answer without access to the gold passages, such as\n"What is the weather in V olcano, CA?" so performance will be lower without using gold passages.\nWe also note that some MSMARCO questions cannot be answered using Wikipedia alone. Here,\nRAG can rely on parametric knowledge to generate reasonable responses.\n3.3 Jeopardy Question Generation\nTo evaluate RAG’s generation abilities in a non-QA setting, we study open-domain question gen-\neration. Rather than use questions from standard open-domain QA tasks, which typically consist\nof short, simple questions, we propose the more demanding task of generating Jeopardy questions.\nJeopardy is an unusual format that consists of trying to guess an entity from a fact about that entity.\nFor example, "The World Cup" is the answer to the question "In 1986 Mexico scored as

```
the  rst'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 4}, page_content='country to host this international sports
competition twice." As Jeopardy questions are precise,\nfactual statements,
generating Jeopardy questions conditioned on their answer entities constitutes
a\nchallenging knowledge-intensive generation task.\nWe use the splits from
SearchQA [ 10], with 100K train, 14K dev, and 27K test examples. As\nthis is a
new task, we train a BART model for comparison. Following [67], we evaluate
using the\nSQuAD-tuned Q-BLEU-1 metric [ 42]. Q-BLEU is a variant of BLEU with a
higher weight for\nmatching entities and has higher correlation with human
judgment for question generation than\nstandard metrics. We also perform two
human evaluations, one to assess generation factuality, and\none for speci city.
We de ne factuality as whether a statement can be corroborated by trusted
external\nsources, and speci city as high mutual dependence between the input
and output [ 33]. We follow'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 4}, page_content='best practice and use pairwise comparative
evaluation [34]. Evaluators are shown an answer and two\ngenerated questions,
one from BART and one from RAG. They are then asked to pick one of
four\noptions-quuestion A is better, question B is better, both are good, or
neither is good.\n3.4 Fact Veri cation\nFEVER [ 56] requires classifying whether
a natural language claim is supported or refuted by\nWikipedia, or whether there
is not enough information to decide. The task requires retrieving\nevidence from
Wikipedia relating to the claim and then reasoning over this evidence to
classify\nwhether the claim is true, false, or unveri able from Wikipedia alone.
FEVER is a retrieval problem\ncoupled with an challenging entailment reasoning
task. It also provides an appropriate testbed for\nexploring the RAG models'
ability to handle classi cation rather than generation. We map FEVER\nclass
labels (supports, refutes, or not enough info) to single output tokens and
directly train with'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 4}, page_content='claim-class pairs. Crucially, unlike most
other approaches to FEVER, we do not use supervision on\nretrieved evidence. In
many real-world applications, retrieval supervision signals aren't available,
and\nmodels that do not require such supervision will be applicable to a wider
range of tasks. We explore\ntwo variants: the standard 3-way classi cation task
(supports/refutes/not enough info) and the 2-way\n(supports/refutes) task
studied in Thorne and Vlachos [57]. In both cases we report label accuracy.\n4
Results\n4.1 Open-domain Question Answering\nTable 1 shows results for RAG along
with state-of-the-art models. On all four open-domain QA\ntasks, RAG sets a new
state of the art (only on the T5-comparable split for TQA). RAG combines\nthe
generation  exibility of the "closed-book" (parametric only) approaches and the
performance of\n"open-book" retrieval-based approaches. Unlike REALM and T5+SSM,
RAG enjoys strong results'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 4}, page_content='without expensive, specialized "salient
span masking" pre-training [20]. It is worth noting that RAG's\nretriever is
```

initialized using DPR's retriever, which uses retrieval supervision on Natural
Questions\nand TriviaQA. RAG compares favourably to the DPR QA system, which
uses a BERT-based "cross-\nencoder" to re-rank documents, along with an
extractive reader. RAG demonstrates that neither a\nre-ranker nor extractive
reader is necessary for state-of-the-art performance.\nThere are several
advantages to generating answers even when it is possible to extract them.
Docu-\nments with clues about the answer but do not contain the answer verbatim
can still contribute towards\na correct answer being generated, which is not
possible with standard extractive approaches, leading\n5'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 5}, page_content='Table 1: Open-Domain QA Test Scores. For
TQA,\nleft column uses the standard test set for Open-\nDomain QA, right column
uses the TQA-Wiki\ntest set. See Appendix D for further details.\nModel NQ TQA
WQ CT\nClosed\nBook\nT5-11B [52] 34.5 - /50.1 37.4 -\nT5-11B+SSM[52] 36.6 -
/60.5 44.7 -\nOpen\nBook\nREALM [20] 40.4 - / - 40.7 46.8\nDPR [26] 41.5 57.9/ -
41.1 50.6\nRAG-Token 44.1 55.2/66.1 45.5 50.0\nRAG-Seq. 44.5 56.8/68.0 45.2
52.2\nTable 2: Generation and classi cation Test Scores.\nMS-MARCO SotA is [4],
FEVER-3 is [68] and\nFEVER-2 is [57] *Uses gold context/evidence.\nBest model
without gold access underlined.\nModel Jeopardy MSMARCO FVR3 FVR2\nB-1 QB-1 R-L
B-1 Label Acc.\nSotA - - 49.8* 49.9* 76.8 92.2 *\nBART 15.1 19.7 38.2 41.6 64.0
81.1\nRAG-Tok. 17.3 22.2 40.1 41.5 72.5 89.5RAG-Seq. 14.7 21.4 40.8 44.2\nto
more effective marginalization over documents. Furthermore, RAG can generate
correct answers\neven when the correct answer is not in any retrieved document,
achieving 11.8% accuracy in such'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 5}, page_content='cases for NQ, where an extractive model
would score 0%.\n4.2 Abstractive Question Answering\nAs shown in Table 2, RAG-
Sequence outperforms BART on Open MS-MARCO NLG by 2.6 Bleu\npoints and 2.6
Rouge-L points. RAG approaches state-of-the-art model performance, which
is\nimpressive given that (i) those models access gold passages with speci c
information required to\ngenerate the reference answer , (ii) many questions are
unanswerable without the gold passages, and\n(iii) not all questions are
answerable from Wikipedia alone. Table 3 shows some generated answers\nfrom our
models. Qualitatively, we nd that RAG models hallucinate less and generate
factually\ncorrect text more often than BART. Later, we also show that RAG
generations are more diverse than\nBART generations (see §4.5).\n4.3 Jeopardy
Question Generation\nTable 2 shows that RAG-Token performs better than RAG-
Sequence on Jeopardy question generation,'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 5}, page_content='with both models outperforming BART on
Q-BLEU-1. 4 shows human evaluation results, over 452\npairs of generations from
BART and RAG-Token. Evaluators indicated that BART was more factual\nthan RAG in
only 7.1% of cases, while RAG was more factual in 42.7% of cases, and both RAG
and\nBART were factual in a further 17% of cases, clearly demonstrating the
effectiveness of RAG on\nthe task over a state-of-the-art generation model.
Evaluators also nd RAG generations to be more\nspeci c by a large margin. Table
3 shows typical generations from each model.\nJeopardy questions often contain

two separate pieces of information, and RAG-Token may perform\nbest because it can generate responses that combine content from several documents. Figure 2 shows\nan example. When generating "Sun", the posterior is high for document 2 which mentions "The\nSun Also Rises". Similarly, document 1 dominates the posterior when "A Farewell to Arms" is'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 5}, page_content='generated. Intriguingly, after the rst token of each book is generated, the document posterior attens.\nThis observation suggests that the generator can complete the titles without depending on speci c\ndocuments. In other words, the model's parametric knowledge is suf cient to complete the titles. We\n nd evidence for this hypothesis by feeding the BART-only baseline with the partial decoding"The\nSun. BART completes the generation "The Sun Also Rises" is a novel by this author of "The Sun\nAlso Rises" indicating the title "The Sun Also Rises" is stored in BART's parameters. Similarly,\nBART will complete the partial decoding "The Sun Also Rises" is a novel by this author of "A\nwith "The Sun Also Rises" is a novel by this author of "A Farewell to Arms". This example shows\nhow parametric and non-parametric memories work together-the non-parametric component helps\nto guide the generation, drawing out speci c knowledge stored in the parametric memory.\n4.4 Fact Veri cation'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 5}, page_content='Table 2 shows our results on FEVER. For 3-way classi cation, RAG scores are within 4.3% of\nstate-of-the-art models, which are complex pipeline systems with domain-speci c architectures and\nsubstantial engineering, trained using intermediate retrieval supervision, which RAG does not require.\n6'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 6}, page_content='Document 1: his works are considered classics of American\nliterature … His wartime experiences formed the basis for his novel\n"A Farewell to Arms"(1929) …\nDocument 2: … artists of the 1920s "Lost Generation" expatriate\ncommunity. His debut novel,"The Sun Also Rises", was published\nin 1926.\nBOS\n"\nTheSunAlso\nR ises\n" is a\nnovel\nby this\nauthor\nof " A\nFarewellto\nArms\n"\nDoc 1\nDoc 2\nDoc 3\nDoc 4\nDoc 5\nFigure 2: RAG-Token document posterior p(zi|x,yi,y-i) for each generated token for input "Hem-\ningway" for Jeopardy generation with 5 retrieved documents. The posterior for document 1 is high\nwhen generating "A Farewell to Arms" and for document 2 when generating "The Sun Also Rises".\nTable 3: Examples from generation tasks. RAG models generate more speci c and factually accurate\nresponses. '?' indicates factually incorrect responses, * indicates partially correct responses.\nTask Input Model Generation\nMS-\nMARCO\nde ne middle\near'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 6}, page_content='de ne middle\near\nBART ?The middle ear is the part of the ear between the middle ear and the nose.\nRAG-T The middle ear is the portion of the ear internal to the eardrum.\nRAG-S The middle ear includes the tympanic cavity and the three ossicles.\nwhat currency\nneeded in\nscotland\nBART The currency needed in Scotland is Pound sterling.\nRAG-T

Pound is the currency needed in Scotland.\nRAG-S The currency needed in Scotland is the pound sterling.\nJeopardy\nQuestion\nGener\n-ation\nWashington\nBART ?This state has the largest number of counties in the U.S.\nRAG-T It's the only U.S. state named for a U.S. president\nRAG-S It's the state where you'll nd Mount Rainier National Park\nThe Divine\nComedy\nBART *This epic poem by Dante is divided into 3 parts: the Inferno, the Purgatorio & the Purgatorio\nRAG-T Dante's "Inferno" is the rst part of this epic poem\nRAG-S This 14th century work is divided into 3 sections: "Inferno", "Purgatorio" & "Paradiso"'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 6}, page_content='For 2-way classi cation, we compare against Thorne and Vlachos [57], who train RoBERTa [35]\nto classify the claim as true or false given the gold evidence sentence. RAG achieves an accuracy\nwithin 2.7% of this model, despite being supplied with only the claim and retrieving its own evidence.\nWe also analyze whether documents retrieved by RAG correspond to documents annotated as gold\nevidence in FEVER. We calculate the overlap in article titles between the topkdocuments retrieved\nby RAG and gold evidence annotations. We nd that the top retrieved document is from a gold article\nin 71% of cases, and a gold article is present in the top 10 retrieved articles in 90% of cases.\n4.5 Additional Results\nGeneration Diversity Section 4.3 shows that RAG models are more factual and speci c than\nBART for Jeopardy question generation. Following recent work on diversity-promoting decoding\n[33, 59, 39], we also investigate generation diversity by calculating the ratio of distinct ngrams to'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 6}, page_content='total ngrams generated by different models. Table 5 shows that RAG-Sequence's generations are\nmore diverse than RAG-Token's, and both are signi cantly more diverse than BART without needing\nany diversity-promoting decoding.\nRetrieval Ablations A key feature of RAG is learning to retrieve relevant information for the task.\nTo assess the effectiveness of the retrieval mechanism, we run ablations where we freeze the retriever\nduring training. As shown in Table 6, learned retrieval improves results for all tasks.\nWe compare RAG's dense retriever to a word overlap-based BM25 retriever [53]. Here, we replace\nRAG's retriever with a xed BM25 system, and use BM25 retrieval scores as logits when calculating\np(z|x). Table 6 shows the results. For FEVER, BM25 performs best, perhaps since FEVER claims are\nheavily entity-centric and thus well-suited for word overlap-based retrieval. Differentiable retrieval\nimproves results on all other tasks, especially for Open-Domain QA, where it is crucial.'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 6}, page_content='Index hot-swapping An advantage of non-parametric memory models like RAG is that knowledge\ncan be easily updated at test time. Parametric-only models like T5 or BART need further training to\nupdate their behavior as the world changes. To demonstrate, we build an index using the DrQA [5]\nWikipedia dump from December 2016 and compare outputs from RAG using this index to the newer\nindex from our main results (December 2018). We prepare a list of 82 world leaders who had changed\n7'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP

Tasks.pdf', 'page': 7}, page_content='Table 4: Human assessments for the
Jeopardy\nQuestion Generation Task.\nFactuality Speci city\nBART better 7.1%
16.8%\nRAG better 42.7% 37.4%\nBoth good 11.7% 11.8%\nBoth poor 17.7% 6.9%\nNo
majority 20.8% 20.1%\nTable 5: Ratio of distinct to total tri-grams
for\ngeneration tasks.\nMSMARCO Jeopardy QGen\nGold 89.6% 90.0%\nBART 70.7%
32.4%\nRAG-Token 77.8% 46.8%\nRAG-Seq. 83.5% 53.8%\nTable 6: Ablations on the
dev set. As FEVER is a classi cation task, both RAG models are
equivalent.\nModel NQ TQA WQ CT Jeopardy-QGen MSMarco FVR-3 FVR-2\nExact Match
B-1 QB-1 R-L B-1 Label Accuracy\nRAG-Token-BM25 29.7 41.5 32.1 33.1 17.5 22.3
55.5 48.4 75.1 91.6RAG-Sequence-BM25 31.8 44.1 36.6 33.8 11.1 19.5 56.5
46.9\nRAG-Token-Frozen 37.8 50.1 37.1 51.1 16.7 21.7 55.9 49.4 72.9 89.4RAG-
Sequence-Frozen 41.2 52.1 41.8 52.6 11.8 19.6 56.7 47.3\nRAG-Token 43.5 54.8
46.5 51.9 17.9 22.6 56.2 49.4 74.5 90.6RAG-Sequence 44.0 55.8 44.9 53.4 15.3
21.5 57.2 47.5'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 7}, page_content='between these dates and use a template
"Who is {position}?" (e.g. "Who is the President of Peru?")\nto query our NQ RAG
model with each index. RAG answers 70% correctly using the 2016 index for\n2016
world leaders and 68% using the 2018 index for 2018 world leaders. Accuracy with
mismatched\nindices is low (12% with the 2018 index and 2016 leaders, 4% with
the 2016 index and 2018 leaders).\nThis shows we can update RAG's world
knowledge by simply replacing its non-parametric memory.\nEffect of Retrieving
more documents Models are trained with either 5 or 10 retrieved
latent\ndocuments, and we do not observe signi cant differences in performance
between them. We have the\n exibility to adjust the number of retrieved
documents at test time, which can affect performance and\nruntime. Figure 3
(left) shows that retrieving more documents at test time monotonically
improves\nOpen-domain QA results for RAG-Sequence, but performance peaks for
RAG-Token at 10 retrieved'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 7}, page_content='documents. Figure 3 (right) shows that
retrieving more documents leads to higher Rouge-L for\nRAG-Token at the expense
of Bleu-1, but the effect is less pronounced for RAG-Sequence.\n10 20 30 40
50\nKR e t r i e v e dD o c s\n39\n40\n41\n42\n43\n44NQ Exact Match RAG-
Tok\nRAG-Seq\n10 20 30 40 50\nKR e t r i e v e dD o c s\n40\n50\n60\n70\n80NQ
Answer Recall @ K\nRAG-Tok\nRAG-Seq\nFixed DPR\nBM25\n10 20 30 40 50\nKR e t r i
e v e dD o c s\n48\n50\n52\n54\n56Bleu-1 / Rouge-L score\nRAG-Tok R-L\nRAG-Tok
B-1\nRAG-Seq R-L\nRAG-Seq B-1\nFigure 3: Left: NQ performance as more documents
are retrieved. Center: Retrieval recall perfor-\nmance in NQ. Right: MS-MARCO
Bleu-1 and Rouge-L as more documents are retrieved.\n5 Related Work\nSingle-Task
Retrieval Prior work has shown that retrieval improves performance across a
variety of\nNLP tasks when considered in isolation. Such tasks include open-
domain question answering [5, 29],\nfact checking [56], fact completion [48],
long-form question answering [12], Wikipedia article'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 7}, page_content='generation [36], dialogue [41, 65, 9,
13], translation [17], and language modeling [19, 27]. Our\nwork uni es

```
previous successes in incorporating retrieval into individual tasks, showing
that a single\nretrieval-based architecture is capable of achieving strong
performance across several tasks.\n8'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 8}, page_content='General-Purpose Architectures for NLP
Prior work on general-purpose architectures for NLP\ntasks has shown great
success without the use of retrieval. A single, pre-trained language model\nhas
been shown to achieve strong performance on various classi cation tasks in the
GLUE bench-\nmarks [60, 61] after  ne-tuning [49, 8]. GPT-2 [50] later showed
that a single, left-to-right, pre-trained\nlanguage model could achieve strong
performance across both discriminative and generative tasks.\nFor further
improvement, BART [32] and T5 [51, 52] propose a single, pre-trained encoder-
decoder\nmodel that leverages bi-directional attention to achieve stronger
performance on discriminative\nand generative tasks. Our work aims to expand the
space of possible tasks with a single, uni ed\narchitecture, by learning a
retrieval module to augment pre-trained, generative language models.\nLearned
Retrieval There is signi cant work on learning to retrieve documents in
information'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 8}, page_content='retrieval, more recently with pre-trained,
neural language models [ 44, 26] similar to ours. Some\nwork optimizes the
retrieval module to aid in a speci c, downstream task such as question
answering,\nusing search [46], reinforcement learning [6, 63, 62], or a latent
variable approach [31, 20] as in our\nwork. These successes leverage different
retrieval-based architectures and optimization techniques to\nachieve strong
performance on a single task, while we show that a single retrieval-based
architecture\ncan be  ne-tuned for strong performance on a variety of
tasks.\nMemory-based Architectures Our document index can be seen as a large
external memory for\nneural networks to attend to, analogous to memory networks
[64, 55]. Concurrent work [14] learns\nto retrieve a trained embedding for each
entity in the input, rather than to retrieve raw text as in our\nwork. Other
work improves the ability of dialog models to generate factual text by attending
over'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 8}, page_content='fact embeddings [15, 13]. A key feature of
our memory is that it is comprised of raw text rather\ndistributed
representations, which makes the memory both (i) human-readable, lending a form
of\ninterpretability to our model, and (ii) human-writable, enabling us to
dynamically update the model's\nmemory by editing the document index. This
approach has also been used in knowledge-intensive\ndialog, where generators
have been conditioned on retrieved text directly, albeit obtained via TF-
IDF\nrather than end-to-end learnt retrieval [9].\nRetrieve-and-Edit approaches
Our method shares some similarities with retrieve-and-edit style\napproaches,
where a similar training input-output pair is retrieved for a given input, and
then edited\nto provide a  nal output. These approaches have proved successful
in a number of domains including\nMachine Translation [ 18, 22] and Semantic
Parsing [21]. Our approach does have several differences,'),
```

```
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 8}, page_content='including less of emphasis on lightly
editing a retrieved item, but on aggregating content from several\npieces of
retrieved content, as well as learning latent retrieval, and retrieving evidence
documents\nrather than related training pairs. This said, RAG techniques may
work well in these settings, and\ncould represent promising future work.\n6
Discussion\nIn this work, we presented hybrid generation models with access to
parametric and non-parametric\nmemory. We showed that our RAG models obtain
state of the art results on open-domain QA. We\nfound that people prefer RAG’s
generation over purely parametric BART,  nding RAG more factual\nand speci c. We
conducted an thorough investigation of the learned retrieval component,
validating\nits effectiveness, and we illustrated how the retrieval index can be
hot-swapped to update the model\nwithout requiring any retraining. In future
work, it may be fruitful to investigate if the two components'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 8}, page_content='can be jointly pre-trained from scratch,
either with a denoising objective similar to BART or some\nanother objective.
Our work opens up new research directions on how parametric and non-
parametric\nmemories interact and how to most effectively combine them, showing
promise in being applied to a\nwide variety of NLP tasks.\n9'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 9}, page_content='Broader Impact\nThis work offers several
positive societal bene ts over previous work: the fact that it is more\nstrongly
grounded in real factual knowledge (in this case Wikipedia) makes it
"hallucinate" less\nwith generations that are more factual, and offers more
control and interpretability. RAG could be\nemployed in a wide variety of
scenarios with direct bene t to society, for example by endowing it\nwith a
medical index and asking it open-domain questions on that topic, or by helping
people be more\neffective at their jobs.\nWith these advantages also come
potential downsides: Wikipedia, or any potential external knowledge\nsource,
will probably never be entirely factual and completely devoid of bias. Since RAG
can be\nemployed as a language model, similar concerns as for GPT-2 [50] are
valid here, although arguably\nto a lesser extent, including that it might be
used to generate abuse, faked or misleading content in'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 9}, page_content='the news or on social media; to
impersonate others; or to automate the production of spam/phishing\ncontent
[54]. Advanced language models may also lead to the automation of various jobs
in the\ncoming decades [16]. In order to mitigate these risks, AI systems could
be employed to  ght against\nmisleading content and automated
spam/phishing.\nAcknowledgments\nThe authors would like to thank the reviewers
for their thoughtful and constructive feedback on this\npaper, as well as
HuggingFace for their help in open-sourcing code to run RAG models. The
authors\nwould also like to thank Kyunghyun Cho and Sewon Min for productive
discussions and advice. EP\nthanks supports from the NSF Graduate Research
Fellowship. PL is supported by the FAIR PhD\nprogram.\nReferences\n[1] Payal
Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu,
```

Rangan\nMajumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina'),
Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 9}, page_content='Stoica, Saurabh Tiwary, and Tong Wang. MS MARCO: A Human Generated MAchine\nReading COmprehension Dataset. arXiv:1611.09268 [cs], November 2016. URL http:\n//arxiv.org/abs/1611.09268. arXiv: 1611.09268.\n[2] Petr Baudiš and Jan Šediv`y. Modeling of the question answering task in the yodaqa system. In\nInternational Conference of the Cross-Language Evaluation Forum for European Languages,\npages 222-228. Springer, 2015. URL https://link.springer.com/chapter/10.1007%\n2F978-3-319-24027-5_20 .\n[3] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic Parsing on Freebase\nfrom Question-Answer Pairs. In Proceedings of the 2013 Conference on Empirical Methods\nin Natural Language Processing, pages 1533-1544, Seattle, Washington, USA, October 2013.\nAssociation for Computational Linguistics. URL http://www.aclweb.org/anthology/\nD13-1160.\n[4] Bin Bi, Chenliang Li, Chen Wu, Ming Yan, and Wei Wang. Palm: Pre-training an autoencod-'),
Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 9}, page_content='ing&autoregressive language model for context-conditioned generation. ArXiv, abs/2004.07159,\n2020. URL https://arxiv.org/abs/2004.07159.\n[5] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading Wikipedia to Answer\nOpen-Domain Questions. In Proceedings of the 55th Annual Meeting of the Association for\nComputational Linguistics (Volume 1: Long Papers), pages 1870-1879, Vancouver, Canada,\nJuly 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1171. URL\nhttps://www.aclweb.org/anthology/P17-1171.\n[6] Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Illia Polosukhin, Alexandre Lacoste, and\nJonathan Berant. Coarse-to- ne question answering for long documents. In Proceedings of the\n55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),\npages 209-220, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi:\n10.18653/v1/P17-1020. URL https://www.aclweb.org/anthology/P17-1020.\n10'),
Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 10}, page_content='[7] Christopher Clark and Matt Gardner. Simple and Effective Multi-Paragraph Reading Compre-\nhension. arXiv:1710.10723 [cs], October 2017. URL http://arxiv.org/abs/1710.10723.\narXiv: 1710.10723.\n[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of\nDeep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Con-\nference of the North American Chapter of the Association for Computational Linguistics: Human\nLanguage Technologies, Volume 1 (Long and Short Papers), pages 4171-4186, Minneapolis,\nMinnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423.\nURL https://www.aclweb.org/anthology/N19-1423.\n[9] Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. Wiz-\nard of wikipedia: Knowledge-powered conversational agents. In International Conference on\nLearning Representations, 2019. URL https://openreview.net/forum?id=r1l73iRqKm.'),

Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 10}, page_content='[10] Matthew Dunn, Levent Sagun, Mike Higgins, V . Ugur Guney, V olkan Cirik, and Kyunghyun\nCho. SearchQA: A New Q&A Dataset Augmented with Context from a Search Engine.\narXiv:1704.05179 [cs], April 2017. URL http://arxiv.org/abs/1704.05179. arXiv:\n1704.05179.\n[11] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In Proceed-\nings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1:\nLong Papers), pages 889-898, Melbourne, Australia, July 2018. Association for Computational\nLinguistics. doi: 10.18653/v1/P18-1082. URL https://www.aclweb.org/anthology/\nP18-1082.\n[12] Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. ELI5:\nLong form question answering. In Proceedings of the 57th Annual Meeting of the Association\nfor Computational Linguistics, pages 3558-3567, Florence, Italy, July 2019. Association for\nComputational Linguistics. doi: 10.18653/v1/P19-1346. URL https://www.aclweb.org/'),
Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 10}, page_content='anthology/P19-1346.\n[13] Angela Fan, Claire Gardent, Chloe Braud, and Antoine Bordes. Augmenting transformers\nwith KNN-based composite memory, 2020. URL https://openreview.net/forum?id=\nH1gx1CNKPH.\n[14] Thibault Févry, Livio Baldini Soares, Nicholas FitzGerald, Eunsol Choi, and Tom Kwiatkowski.\nEntities as experts: Sparse memory access with entity supervision. ArXiv, abs/2004.07202,\n2020. URL https://arxiv.org/abs/2004.07202.\n[15] Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen\ntau Yih, and Michel Galley. A knowledge-grounded neural conversation model. In AAAI\nConference on Arti cial Intelligence, 2018. URL https://www.aaai.org/ocs/index.php/\nAAAI/AAAI18/paper/view/16710.\n[16] Katja Grace, John Salvatier, Allan Dafoe, Baobao Zhang, and Owain Evans. When will AI\nexceed human performance? evidence from AI experts. CoRR, abs/1705.08807, 2017. URL\nhttp://arxiv.org/abs/1705.08807.'),
Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 10}, page_content='[17] Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor O.K. Li. Search engine guided neural\nmachine translation. In AAAI Conference on Arti cial Intelligence , 2018. URL https:\n//www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17282.\n[18] Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor O.K. Li. Search engine guided neural\nmachine translation. In 32nd AAAI Conference on Arti cial Intelligence, AAAI 2018 , 32nd\nAAAI Conference on Arti cial Intelligence, AAAI 2018, pages 5133-5140. AAAI press, 2018.\n32nd AAAI Conference on Arti cial Intelligence, AAAI 2018 ; Conference date: 02-02-2018\nThrough 07-02-2018.\n[19] Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. Generating sentences by\nediting prototypes. Transactions of the Association for Computational Linguistics, 6:437-450,\n2018. doi: 10.1162/tacl_a_00030. URL https://www.aclweb.org/anthology/Q18-1031.\n11'),
Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 11}, page_content='[20] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. REALM:\nRetrieval-augmented language model

pre-training. ArXiv, abs/2002.08909, 2020. URL
https:\n//arxiv.org/abs/2002.08909.\n[21] Tatsunori B Hashimoto, Kelvin Guu, Yonatan Oren, and Percy S Liang. A\nretrieve-and-edit framework for predicting structured outputs. In S. Bengio,\nH. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, ed-\nitors, Advances in Neural Information Processing Systems 31 , pages 10052-\n10062. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/\n8209-a-retrieve-and-edit-framework-for-predicting-structured-outputs.\npdf.\n[22] Nabil Hossain, Marjan Ghazvininejad, and Luke Zettlemoyer. Simple and effective retrieve-\nedit-rerank text generation. In Proceedings of the 58th Annual Meeting of the Association for\nComputational Linguistics, pages 2532-2538, Online, July 2020. Association for Computa-'),
Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 11}, page_content='tional Linguistics. doi: 10.18653/v1/2020.acl-main.228. URL https://www.aclweb.org/\nanthology/2020.acl-main.228.\n[23] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. arXiv\npreprint arXiv:1702.08734, 2017. URL https://arxiv.org/abs/1702.08734.\n[24] Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A Large Scale\nDistantly Supervised Challenge Dataset for Reading Comprehension. In Proceedings of the\n55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),\npages 1601-1611, Vancouver, Canada, July 2017. Association for Computational Linguistics.\ndoi: 10.18653/v1/P17-1147. URL
https://www.aclweb.org/anthology/P17-1147.\n[25] Armand Joulin and Tomas Mikolov. Inferring algorithmic patterns with stack-\naugmented recurrent nets. In Proceedings of the 28th International Conference on\nNeural Information Processing Systems - Volume 1 , NIPS’15, page 190-198, Cam-'),
Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 11}, page_content='bridge, MA, USA, 2015. MIT Press. URL https://papers.nips.cc/paper/\n5857-inferring-algorithmic-patterns-with-stack-augmented-recurrent-nets .\n[26] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and\nWen-tau Yih. Dense passage retrieval for open-domain question answering. arXiv preprint\narXiv:2004.04906, 2020. URL https://arxiv.org/abs/2004.04906.\n[27] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generaliza-\ntion through memorization: Nearest neighbor language models. In International Conference on\nLearning Representations, 2020. URL
https://openreview.net/forum?id=HklBjCEKvH.\n[28] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua\nBengio and Yann LeCun, editors, 3rd International Conference on Learning Representations,\nICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015. URL\nhttp://arxiv.org/abs/1412.6980.'),
Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 11}, page_content='[29] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Red eld, Michael Collins, Ankur Parikh,\nChris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Ken-\nton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob\nUszkoreit, Quoc Le, and Slav Petrov. Natural Questions: a Benchmark for

Ques-\ntion Answering Research. Transactions of the Association of Computational Lin-\nguistics, 2019. URL https://tomkwiat.users.x20web.corp.google.com/papers/\nnatural-questions/main-1455-kwiatkowski.pdf .\n[30] Guillaume Lample, Alexandre Sablayrolles, Marc’ Aurelio Ranzato, Ludovic Denoyer, and\nHerve Jegou. Large memory layers with product keys. In H. Wallach, H. Larochelle,\nA. Beygelzimer, F. d’ Alché-Buc, E. Fox, and R. Garnett, editors, Advances in Neural In-\nformation Processing Systems 32, pages 8548-8559. Curran Associates, Inc., 2019. URL http:\n//papers.nips.cc/paper/9061-large-memory-layers-with-product-keys.pdf .'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 11}, page_content='[31] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised\nopen domain question answering. In Proceedings of the 57th Annual Meeting of the Association\n12'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 12}, page_content='for Computational Linguistics, pages 6086-6096, Florence, Italy, July 2019. Association for\nComputational Linguistics. doi: 10.18653/v1/P19-1612. URL https://www.aclweb.org/\nanthology/P19-1612.\n[32] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed,\nOmer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence\npre-training for natural language generation, translation, and comprehension. arXiv preprint\narXiv:1910.13461, 2019. URL https://arxiv.org/abs/1910.13461.\n[33] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A diversity-promoting\nobjective function for neural conversation models. In Proceedings of the 2016 Conference of the\nNorth American Chapter of the Association for Computational Linguistics: Human Language\nTechnologies, pages 110-119, San Diego, California, June 2016. Association for Computational\nLinguistics. doi: 10.18653/v1/N16-1014. URL https://www.aclweb.org/anthology/\nN16-1014.'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 12}, page_content='N16-1014.\n[34] Margaret Li, Jason Weston, and Stephen Roller. Acute-eval: Improved dialogue evaluation\nwith optimized questions and multi-turn comparisons. ArXiv, abs/1909.03087, 2019. URL\nhttps://arxiv.org/abs/1909.03087.\n[35] Hairong Liu, Mingbo Ma, Liang Huang, Hao Xiong, and Zhongjun He. Robust neural machine\ntranslation with joint textual and phonetic embedding. In Proceedings of the 57th Annual\nMeeting of the Association for Computational Linguistics, pages 3044-3049, Florence, Italy,\nJuly 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1291. URL\nhttps://www.aclweb.org/anthology/P19-1291.\n[36] Peter J. Liu*, Mohammad Saleh*, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser,\nand Noam Shazeer. Generating wikipedia by summarizing long sequences. In International\nConference on Learning Representations, 2018. URL https://openreview.net/forum?\nid=Hyg0vbWC-.\n[37] Yury A. Malkov and D. A. Yashunin. Ef cient and robust approximate nearest neighbor search'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 12}, page_content='using hierarchical navigable small world

graphs. IEEE Transactions on Pattern Analysis and\nMachine Intelligence,
42:824-836, 2016. URL https://arxiv.org/abs/1603.09320.\n[38] Gary Marcus. The
next decade in ai: four steps towards robust arti cial intelligence.
arXiv\npreprint arXiv:2002.06177, 2020. URL
https://arxiv.org/abs/2002.06177.\n[39] Luca Massarelli, Fabio Petroni,
Aleksandra Piktus, Myle Ott, Tim Rocktäschel, Vassilis\nPlachouras, Fabrizio
Silvestri, and Sebastian Riedel. How decoding strategies affect
the\nveri ability of generated text. arXiv preprint arXiv:1911.03587 , 2019. URL
https:\n//arxiv.org/abs/1911.03587.\n[40] Paulius Micikevicius, Sharan Narang,
Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia,\nBoris Ginsburg, Michael
Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed\nprecision
training. In ICLR, 2018. URL https://openreview.net/forum?id=r1gs9JgRZ.\n[41]
Nikita Moghe, Siddhartha Arora, Suman Banerjee, and Mitesh M. Khapra. Towards
exploit-'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 12}, page_content='ing background knowledge for building
conversation systems. In Proceedings of the 2018\nConference on Empirical
Methods in Natural Language Processing, pages 2322-2332, Brus-\nsels, Belgium,
October-November 2018. Association for Computational Linguistics.
doi:\n10.18653/v1/D18-1255. URL https://www.aclweb.org/anthology/D18-1255.\n[42]
Preksha Nema and Mitesh M. Khapra. Towards a better metric for evaluating
question generation\nsystems. In Proceedings of the 2018 Conference on Empirical
Methods in Natural Language\nProcessing, pages 3950-3959, Brussels, Belgium,
October-November 2018. Association for\nComputational Linguistics. doi:
10.18653/v1/D18-1429. URL https://www.aclweb.org/\nanthology/D18-1429.\n[43] Tri
Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan
Majumder,\nand Li Deng. MS MARCO: A human generated machine reading
comprehension dataset. In\nTarek Richard Besold, Antoine Bordes, Artur S.
d'Avila Garcez, and Greg Wayne, editors,'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 12}, page_content='Proceedings of the Workshop on Cognitive
Computation: Integrating neural and symbolic\n13'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 13}, page_content='approaches 2016 co-located with the 30th
Annual Conference on Neural Information Processing\nSystems (NIPS 2016),
Barcelona, Spain, December 9, 2016, volume 1773 of CEUR Workshop\nProceedings.
CEUR-WS.org, 2016. URL http://ceur-
ws.org/Vol-1773/CoCoNIPS_\n2016_paper9.pdf.\n[44] Rodrigo Nogueira and Kyunghyun
Cho. Passage re-ranking with BERT. arXiv preprint\narXiv:1901.04085, 2019. URL
https://arxiv.org/abs/1901.04085.\n[45] Myle Ott, Sergey Edunov, Alexei Baevski,
Angela Fan, Sam Gross, Nathan Ng, David Grangier,\nand Michael Auli. fairseq: A
fast, extensible toolkit for sequence modeling. In Proceedings\nof the 2019
Conference of the North American Chapter of the Association for
Computational\nLinguistics (Demonstrations), pages 48-53, Minneapolis,
Minnesota, June 2019. Association\nfor Computational Linguistics. doi:
10.18653/v1/N19-4009. URL https://www.aclweb.\norg/anthology/N19-4009.\n[46]
Ethan Perez, Siddharth Karamcheti, Rob Fergus, Jason Weston, Douwe Kiela, and

```
Kyunghyun'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 13}, page_content='Cho. Finding generalizable evidence by
learning to convince q&a models. In Proceedings\nof the 2019 Conference on
Empirical Methods in Natural Language Processing and the 9th\nInternational
Joint Conference on Natural Language Processing (EMNLP-IJCNLP) ,
pages\n2402–2411, Hong Kong, China, November 2019. Association for Computational
Linguistics.\ndoi: 10.18653/v1/D19-1244. URL
https://www.aclweb.org/anthology/D19-1244.\n[47] Fabio Petroni, Tim Rocktäschel,
Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu,\nand Alexander
Miller. Language models as knowledge bases? In Proceedings of the
2019\nConference on Empirical Methods in Natural Language Processing and the 9th
International\nJoint Conference on Natural Language Processing (EMNLP-IJCNLP),
pages 2463–2473, Hong\nKong, China, November 2019. Association for Computational
Linguistics. doi: 10.18653/v1/\nD19-1250. URL
https://www.aclweb.org/anthology/D19-1250.'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 13}, page_content='[48] Fabio Petroni, Patrick Lewis,
Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H.\nMiller, and
Sebastian Riedel. How context affects language models' factual predictions.
In\nAutomated Knowledge Base Construction, 2020. URL
https://openreview.net/forum?\nid=025X0zPfn.\n[49] Alec Radford, Karthik
Narasimhan, Tim Salimans, and Ilya Sutskever. Im-\nproving Language
Understanding by Generative Pre-Training, 2018. URL\nhttps://s3-us-
west-2.amazonaws.com/openai-assets/research-covers/\nlanguage-
unsupervised/language_understanding_paper.pdf.\n[50] Alec Radford, Jeff Wu,
Rewon Child, David Luan, Dario Amodei, and Ilya\nSutskever. Language models are
unsupervised multitask learners, 2019.
URL\nhttps://d4mucfpksywv.cloudfront.net/better-language-
models/language_\nmodels_are_unsupervised_multitask_learners.pdf.\n[51] Colin
Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael
Matena,'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 13}, page_content='Yanqi Zhou, Wei Li, and Peter J. Liu.
Exploring the limits of transfer learning with a uni ed\ntext-to-text
transformer. arXiv e-prints, 2019. URL https://arxiv.org/abs/1910.10683.\n[52]
Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack
into\nthe parameters of a language model? arXiv e-prints, 2020. URL
https://arxiv.org/abs/\n2002.08910.\n[53] Stephen Robertson and Hugo Zaragoza.
The probabilistic relevance framework: Bm25 and\nbeyond. Found. Trends Inf.
Retr., 3(4):333–389, April 2009. ISSN 1554-0669. doi: 10.1561/\n1500000019. URL
https://doi.org/10.1561/1500000019.\n[54] Irene Solaiman, Miles Brundage, Jack
Clark, Amanda Askell, Ariel Herbert-V oss, Jeff Wu, Alec\nRadford, and Jian-Bing
Wang. Release strategies and the social impacts of language models.\nArXiv,
abs/1908.09203, 2019.\n[55] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston,
and Rob Fergus. End-to-end memory net-'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
```

Tasks.pdf', 'page': 13}, page_content='works. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors,Advances\nin Neural Information Processing Systems 28, pages 2440-2448. Curran Associates, Inc., 2015.\nURL http://papers.nips.cc/paper/5846-end-to-end-memory-networks.pdf .\n14'),
Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 14}, page_content='[56] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a\nlarge-scale dataset for fact extraction and VERi cation. In Proceedings of the 2018 Conference\nof the North American Chapter of the Association for Computational Linguistics: Human\nLanguage Technologies, Volume 1 (Long Papers), pages 809-819, New Orleans, Louisiana,\nJune 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1074. URL\nhttps://www.aclweb.org/anthology/N18-1074.\n[57] James H. Thorne and Andreas Vlachos. Avoiding catastrophic forgetting in mitigating model\nbiases in sentence-pair classi cation with elastic weight consolidation. ArXiv, abs/2004.14366,\n2020. URL https://arxiv.org/abs/2004.14366.\n[58] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez,\nŁ ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V . Luxburg,'),
Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 14}, page_content='S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors,Advances in Neural\nInformation Processing Systems 30, pages 5998-6008. Curran Associates, Inc., 2017. URL\nhttp://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf .\n[59] Ashwin Vijayakumar, Michael Cogswell, Ramprasaath Selvaraju, Qing Sun, Stefan Lee, David\nCrandall, and Dhruv Batra. Diverse beam search for improved description of complex scenes.\nAAAI Conference on Arti cial Intelligence, 2018. URL https://www.aaai.org/ocs/index.\nphp/AAAI/AAAI18/paper/view/17329.\n[60] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman.\nGLUE: A multi-task benchmark and analysis platform for natural language understanding.\nIn Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting\nNeural Networks for NLP, pages 353-355, Brussels, Belgium, November 2018. Association for\nComputational Linguistics. doi: 10.18653/v1/W18-5446. URL https://www.aclweb.org/'),
Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 14}, page_content='anthology/W18-5446.\n[61] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix\nHill, Omer Levy, and Samuel Bowman. SuperGLUE: A Stickier Benchmark for General-\nPurpose Language Understanding Systems. In H. Wallach, H. Larochelle, A. Beygelzimer,\nF. d\\textquotesingle Alché-Buc, E. Fox, and R. Garnett, editors,Advances in Neural Information\nProcessing Systems 32, pages 3261-3275. Curran Associates, Inc., 2019. URL https://\narxiv.org/abs/1905.00537.\n[62] Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang,\nGerry Tesauro, Bowen Zhou, and Jing Jiang. R3: Reinforced ranker-reader for open-domain\nquestion answering. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, Proceedings of\nthe Thirty-Second AAAI Conference on Arti cial Intelligence, (AAAI-18), the 30th innovative\nApplications of Arti cial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational'),

Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 14}, page_content='Advances in Arti cial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7,\n2018, pages 5981-5988. AAAI Press, 2018. URL https://www.aaai.org/ocs/index.\nphp/AAAI/AAAI18/paper/view/16712.\n[63] Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang,\nTim Klinger, Gerald Tesauro, and Murray Campbell. Evidence aggregation for answer re-\nranking in open-domain question answering. In ICLR, 2018. URL https://openreview.\nnet/forum?id=rJl3yM-Ab.\n[64] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. In Yoshua Bengio\nand Yann LeCun, editors, 3rd International Conference on Learning Representations, ICLR\n2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings , 2015. URL\nhttp://arxiv.org/abs/1410.3916.\n[65] Jason Weston, Emily Dinan, and Alexander Miller. Retrieve and re ne: Improved sequence\ngeneration models for dialogue. In Proceedings of the 2018 EMNLP Workshop SCAI: The 2nd'),
Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 14}, page_content='International Workshop on Search-Oriented Conversational AI, pages 87-92, Brussels, Belgium,\nOctober 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5713. URL\nhttps://www.aclweb.org/anthology/W18-5713.\n15'),
Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 15}, page_content='[66] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony\nMoi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer,\nPatrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain\nGugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface’s transformers:\nState-of-the-art natural language processing. ArXiv, abs/1910.03771, 2019.\n[67] Shiyue Zhang and Mohit Bansal. Addressing semantic drift in question generation for semi-\nsupervised question answering. In Proceedings of the 2019 Conference on Empirical Meth-\nods in Natural Language Processing and the 9th International Joint Conference on Natural\nLanguage Processing (EMNLP-IJCNLP) , pages 2495-2509, Hong Kong, China, Novem-\nber 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1253. URL\nhttps://www.aclweb.org/anthology/D19-1253.'),
Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 15}, page_content='[68] Wanjun Zhong, Jingjing Xu, Duyu Tang, Zenan Xu, Nan Duan, Ming Zhou, Jiahai Wang, and\nJian Yin. Reasoning over semantic-level graph for fact checking. ArXiv, abs/1909.03745, 2019.\nURL https://arxiv.org/abs/1909.03745.\n16'),
Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 16}, page_content='Appendices for Retrieval-Augmented Generation for\nKnowledge-Intensive NLP Tasks\nA Implementation Details\nFor Open-domain QA we report test numbers using 15 retrieved documents for RAG-Token models.\nFor RAG-Sequence models, we report test results using 50 retrieved documents, and we use the\nThorough Decoding approach since answers are generally short. We use greedy decoding for QA as\nwe did not nd beam search improved results. For Open-MSMarco and Jeopardy question generation,\nwe report

test numbers using ten retrieved documents for both RAG-Token and RAG-Sequence,\nand we also train a BART-large model as a baseline. We use a beam size of four, and use the Fast\nDecoding approach for RAG-Sequence models, as Thorough Decoding did not improve performance.\nB Human Evaluation\nFigure 4: Annotation interface for human evaluation of factuality. A pop-out for detailed instructions\nand a worked example appear when clicking "view tool guide".'),
Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 16}, page_content='Figure 4 shows the user interface for human evaluation. To avoid any biases for screen position,\nwhich model corresponded to sentence A and sentence B was randomly selected for each example.\nAnnotators were encouraged to research the topic using the internet, and were given detailed instruc-\ntions and worked examples in a full instructions tab. We included some gold sentences in order to\nassess the accuracy of the annotators. Two annotators did not perform well on these examples and\ntheir annotations were removed from the results.\nC Training setup Details\nWe train all RAG models and BART baselines using Fairseq [45].2 We train with mixed precision\n oating point arithmetic [40], distributing training across 8, 32GB NVIDIA V100 GPUs, though\ntraining and inference can be run on one GPU. We nd that doing Maximum Inner Product Search\nwith FAISS is suf ciently fast on CPU, so we store document index vectors on CPU, requiring 100'),
Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 16}, page_content='GB of CPU memory for all of Wikipedia. After submission, We have ported our code to HuggingFace\nTransformers [66]3, which achieves equivalent performance to the previous version but is a cleaner\nand easier to use implementation. This version is also open-sourced. We also compress the document\nindex using FAISS's compression tools, reducing the CPU memory requirement to 36GB. Scripts to\nrun experiments with RAG can be found athttps://github.com/huggingface/transformers/\nblob/master/examples/rag/README.md and an interactive demo of a RAG model can be found\nat https://huggingface.co/rag/\n2https://github.com/pytorch/fairseq\n3https://github.com/huggingface/transformers\n17Viewfull instructions\nWhich sentence is more factually true?\nView tool guide\nSelect an option\nSubject\n: Hemingway\nNote:Somequestionsare\nSentence A is more\n1\ncontrol questions.Werequire\ntrue\nSentence A: "The Sun Also Rises" is a novel by this author of "A\ngood accuracy on our control\nFarewell to Arms"\nSentenceBismore\n2'),
Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 16}, page_content='SentenceBismore\n2\nquestions to accept\ntrue\nresponses.\nSentenceB:Thisauthorof"TheSunAlsoRises"wasbornin\nBoth sentences are\n3\nHavana, Cuba, the son of Spanish immigrants\ntrue\nIndicatewhichoneofthe\nfollowing sentences is more\nBoth sentences are\ncompletely untrue\nfactuallytruewithrespectto\nthe subject. Using the\ninternet to check whether\nthesentencesaretrueis\nencouraged.'),
Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 17}, page_content='D Further Details on Open-Domain QA\nFor open-domain QA, multiple answer annotations are often available for a given

question. These\nanswer annotations are exploited by extractive models during training as typically all the answer\nannotations are used to nd matches within documents when preparing training data. For RAG, we\nalso make use of multiple annotation examples for Natural Questions and WebQuestions by training\nthe model with each (q,a) pair separately, leading to a small increase in accuracy. For TriviaQA,\nthere are often many valid answers to a given question, some of which are not suitable training targets,\nsuch as emoji or spelling variants. For TriviaQA, we lter out answer candidates if they do not occur\nin top 1000 documents for the query.\nCuratedTrec preprocessing The answers for CuratedTrec are given in the form of regular expres-\nsions, which has been suggested as a reason why it is unsuitable for answer-generation models [20].'),

Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 17}, page_content='To overcome this, we use a pre-processing step where we rst retrieve the top 1000 documents for\neach query, and use the answer that most frequently matches the regex pattern as the supervision\ntarget. If no matches are found, we resort to a simple heuristic: generate all possible permutations for\neach regex, replacing non-deterministic symbols in the regex nested tree structure with a whitespace.\nTriviaQA Evaluation setups The open-domain QA community customarily uses public develop-\nment datasets as test datasets, as test data for QA datasets is often restricted and dedicated to reading\ncompehension purposes. We report our results using the datasets splits used in DPR [26], which are\nconsistent with common practice in Open-domain QA. For TriviaQA, this test dataset is the public\nTriviaQA Web Development split. Roberts et al.[52] used the TriviaQA of cial Wikipedia test set\ninstead. Févry et al. [14] follow this convention in order to compare with Roberts et al. [52] (See'),

Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 17}, page_content='appendix of [14]). We report results on both test sets to enable fair comparison to both approaches.\nWe nd that our performance is much higher using the of cial Wiki test set, rather than the more\nconventional open-domain test set, which we attribute to the of cial Wiki test set questions being\nsimpler to answer from Wikipedia.\nE Further Details on FEVER\nFor FEVER classi cation, we follow the practice from [32], and rst re-generate the claim, and\nthen classify using the representation of the nal hidden state, before nally marginalizing across\ndocuments to obtain the class probabilities. The FEVER task traditionally has two sub-tasks. The\n rst is to classify the claim as either "Supported", "Refuted" or "Not Enough Info", which is the task\nwe explore in the main paper. FEVER's other sub-task involves extracting sentences from Wikipedia\nas evidence supporting the classi cation prediction. As FEVER uses a different Wikipedia dump to'),

Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP Tasks.pdf', 'page': 17}, page_content='us, directly tackling this task is not straightforward. We hope to address this in future work.\nF Null Document Probabilities\nWe experimented with adding "Null document" mechanism to RAG, similar to REALM [20] in order\nto model cases where no useful information could be retrieved for a given input. Here, ifkdocuments\nwere retrieved, we would additionally "retrieve" an empty document and predict a logit for the

```
null\ndocument, before marginalizing over k+ 1predictions. We explored modelling
this null document\nlogit by learning (i) a document embedding for the null
document, (ii) a static learnt bias term, or\n(iii) a neural network to predict
the logit. We did not  nd that these improved performance, so in\nthe interests
of simplicity, we omit them. For Open MS-MARCO, where useful retrieved
documents\ncannot always be retrieved, we observe that the model learns to
always retrieve a particular set of'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 17}, page_content='documents for questions that are less
likely to bene t from retrieval, suggesting that null document\nmechanisms may
not be necessary for RAG.\nG Parameters\nOur RAG models contain the trainable
parameters for the BERT-base query and document encoder of\nDPR, with 110M
parameters each (although we do not train the document encoder ourselves)
and\n406M trainable parameters from BART-large, 406M parameters, making a total
of 626M trainable\n18'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 18}, page_content='Table 7: Number of instances in the
datasets used. *A hidden subset of this data is used for evaluation\nTask Train
Development Test\nNatural Questions 79169 8758 3611\nTriviaQA 78786 8838
11314\nWebQuestions 3418 362 2033\nCuratedTrec 635 134 635\nJeopardy Question
Generation 97392 13714 26849\nMS-MARCO 153726 12468 101093*\nFEVER-3-way 145450
10000 10000\nFEVER-2-way 96966 6666 6666\nparameters. The best performing
"closed-book" (parametric only) open-domain QA model is T5-11B\nwith 11 Billion
trainable parameters. The T5 model with the closest number of parameters to
our\nmodels is T5-large (770M parameters), which achieves a score of 28.9 EM on
Natural Questions [52],\nsubstantially below the 44.5 that RAG-Sequence
achieves, indicating that hybrid parametric/non-\nparametric models require far
fewer trainable parameters for strong open-domain QA performance.\nThe non-
parametric memory index does not consist of trainable parameters, but does
consists of 21M'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 18}, page_content='728 dimensional vectors, consisting of
15.3B values. These can be easily be stored at 8-bit  oating\npoint precision to
manage memory and disk footprints.\nH Retrieval Collapse\nIn preliminary
experiments, we observed that for some tasks such as story generation [ 11],
the\nretrieval component would "collapse" and learn to retrieve the same
documents regardless of the\ninput. In these cases, once retrieval had
collapsed, the generator would learn to ignore the documents,\nand the RAG model
would perform equivalently to BART. The collapse could be due to a less-
explicit\nrequirement for factual knowledge in some tasks, or the longer target
sequences, which could result\nin less informative gradients for the retriever.
Perez et al.[46] also found spurious retrieval results\nwhen optimizing a
retrieval component in order to improve performance on downstream tasks.\nI
Number of instances per dataset\nThe number of training, development and test
datapoints in each of our datasets is shown in Table 7.'),
 Document(metadata={'source': '/content/Retrieval-Augmented Generation for NLP
Tasks.pdf', 'page': 18}, page_content='19')]
```

[26]:
```
from langchain.vectorstores import Weaviate
# Initialize the Weaviate vectorstore, providing the weaviate_url
vector_db = Weaviate.from_documents(
    docs,
    embeddings,
    client=client,  # Pass the Weaviate URL here
    by_text=False
)
```

[27]:
```
print(vector_db)
```

```
<langchain_community.vectorstores.weaviate.Weaviate object at 0x7f108fe358a0>
```

[28]:
```
vector_db.similarity_search("What is attention?", k=3)
```

[28]:
```
[Document(metadata={'page': 14, 'source': '/content/Retrieval-Augmented
Generation for NLP Tasks.pdf'}, page_content='S. Bengio, H. Wallach, R. Fergus,
S. Vishwanathan, and R. Garnett, editors,Advances in Neural\nInformation
Processing Systems 30, pages 5998-6008. Curran Associates, Inc., 2017.
URL\nhttp://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf .\n[59]
Ashwin Vijayakumar, Michael Cogswell, Ramprasaath Selvaraju, Qing Sun, Stefan
Lee, David\nCrandall, and Dhruv Batra. Diverse beam search for improved
description of complex scenes.\nAAAI Conference on Arti cial Intelligence, 2018.
URL https://www.aaai.org/ocs/index.\nphp/AAAI/AAAI18/paper/view/17329.\n[60]
Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel
Bowman.\nGLUE: A multi-task benchmark and analysis platform for natural language
understanding.\nIn Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing
and Interpreting\nNeural Networks for NLP, pages 353-355, Brussels, Belgium,
November 2018. Association for\nComputational Linguistics. doi:
10.18653/v1/W18-5446. URL https://www.aclweb.org/'),
 Document(metadata={'page': 14, 'source': '/content/Retrieval-Augmented
Generation for NLP Tasks.pdf'}, page_content='[56] James Thorne, Andreas
Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a\nlarge-scale
dataset for fact extraction and VERi cation. In Proceedings of the 2018
Conference\nof the North American Chapter of the Association for Computational
Linguistics: Human\nLanguage Technologies, Volume 1 (Long Papers), pages
809-819, New Orleans, Louisiana,\nJune 2018. Association for Computational
Linguistics. doi: 10.18653/v1/N18-1074.
URL\nhttps://www.aclweb.org/anthology/N18-1074.\n[57] James H. Thorne and
Andreas Vlachos. Avoiding catastrophic forgetting in mitigating model\nbiases in
sentence-pair classi cation with elastic weight consolidation. ArXiv,
abs/2004.14366,\n2020. URL https://arxiv.org/abs/2004.14366.\n[58] Ashish
Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N
Gomez,\nŁ ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I.
Guyon, U. V . Luxburg,'),
 Document(metadata={'page': 8, 'source': '/content/Retrieval-Augmented
Generation for NLP Tasks.pdf'}, page_content='General-Purpose Architectures for
NLP Prior work on general-purpose architectures for NLP\ntasks has shown great
```

```
success without the use of retrieval. A single, pre-trained language model\nhas
been shown to achieve strong performance on various classi cation tasks in the
GLUE bench-\nmarks [60, 61] after  ne-tuning [49, 8]. GPT-2 [50] later showed
that a single, left-to-right, pre-trained\nlanguage model could achieve strong
performance across both discriminative and generative tasks.\nFor further
improvement, BART [32] and T5 [51, 52] propose a single, pre-trained encoder-
decoder\nmodel that leverages bi-directional attention to achieve stronger
performance on discriminative\nand generative tasks. Our work aims to expand the
space of possible tasks with a single, uni ed\narchitecture, by learning a
retrieval module to augment pre-trained, generative language models.\nLearned
Retrieval There is signi cant work on learning to retrieve documents in
information')]
```

[31]:
```python
print(vector_db.similarity_search("What is attention?", k=3)[0].page_content)
```

```
S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett,
editors,Advances in Neural
Information Processing Systems 30, pages 5998-6008. Curran Associates, Inc.,
2017. URL
http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf .
[59] Ashwin Vijayakumar, Michael Cogswell, Ramprasaath Selvaraju, Qing Sun,
Stefan Lee, David
Crandall, and Dhruv Batra. Diverse beam search for improved description of
complex scenes.
AAAI Conference on Arti cial Intelligence, 2018. URL
https://www.aaai.org/ocs/index.
php/AAAI/AAAI18/paper/view/17329.
[60] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and
Samuel Bowman.
GLUE: A multi-task benchmark and analysis platform for natural language
understanding.
In Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and
Interpreting
Neural Networks for NLP, pages 353-355, Brussels, Belgium, November 2018.
Association for
Computational Linguistics. doi: 10.18653/v1/W18-5446. URL
https://www.aclweb.org/
```

[32]:
```python
print(vector_db.similarity_search("What is attention?", k=3)[2].page_content)
```

```
General-Purpose Architectures for NLP Prior work on general-purpose
architectures for NLP
tasks has shown great success without the use of retrieval. A single, pre-
trained language model
has been shown to achieve strong performance on various classi cation tasks in
the GLUE bench-
marks [60, 61] after  ne-tuning [49, 8]. GPT-2 [50] later showed that a single,
left-to-right, pre-trained
```

```
language model could achieve strong performance across both discriminative and
generative tasks.
For further improvement, BART [32] and T5 [51, 52] propose a single, pre-trained
encoder-decoder
model that leverages bi-directional attention to achieve stronger performance on
discriminative
and generative tasks. Our work aims to expand the space of possible tasks with a
single, uni ed
architecture, by learning a retrieval module to augment pre-trained, generative
language models.
Learned Retrieval There is signi cant work on learning to retrieve documents in
information
```

[33]:
```
print(vector_db.similarity_search("What is attention?", k=3)[1].page_content)
```

```
[56] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit
Mittal. FEVER: a
large-scale dataset for fact extraction and VERi cation. In Proceedings of the
2018 Conference
of the North American Chapter of the Association for Computational Linguistics:
Human
Language Technologies, Volume 1 (Long Papers), pages 809-819, New Orleans,
Louisiana,
June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1074.
URL
https://www.aclweb.org/anthology/N18-1074.
[57] James H. Thorne and Andreas Vlachos. Avoiding catastrophic forgetting in
mitigating model
biases in sentence-pair classi cation with elastic weight consolidation. ArXiv,
abs/2004.14366,
2020. URL https://arxiv.org/abs/2004.14366.
[58] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones,
Aidan N Gomez,
Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U.
V . Luxburg,
```

[34]:
```
vector_db.similarity_search("What is RAG?", k=3)
```

[34]:
```
[Document(metadata={'page': 17, 'source': '/content/Retrieval-Augmented
Generation for NLP Tasks.pdf'}, page_content='documents for questions that are
less likely to bene t from retrieval, suggesting that null document\nmechanisms
may not be necessary for RAG.\nG Parameters\nOur RAG models contain the
trainable parameters for the BERT-base query and document encoder of\nDPR, with
110M parameters each (although we do not train the document encoder ourselves)
and\n406M trainable parameters from BART-large, 406M parameters, making a total
of 626M trainable\n18'),
 Document(metadata={'page': 5, 'source': '/content/Retrieval-Augmented
Generation for NLP Tasks.pdf'}, page_content='cases for NQ, where an extractive
```

```
model would score 0%.\n4.2 Abstractive Question Answering\nAs shown in Table 2,
RAG-Sequence outperforms BART on Open MS-MARCO NLG by 2.6 Bleu\npoints and 2.6
Rouge-L points. RAG approaches state-of-the-art model performance, which
is\nimpressive given that (i) those models access gold passages with speci c
information required to\ngenerate the reference answer , (ii) many questions are
unanswerable without the gold passages, and\n(iii) not all questions are
answerable from Wikipedia alone. Table 3 shows some generated answers\nfrom our
models. Qualitatively, we  nd that RAG models hallucinate less and generate
factually\ncorrect text more often than BART. Later, we also show that RAG
generations are more diverse than\nBART generations (see §4.5).\n4.3 Jeopardy
Question Generation\nTable 2 shows that RAG-Token performs better than RAG-
Sequence on Jeopardy question generation,'),
 Document(metadata={'page': 9, 'source': '/content/Retrieval-Augmented
Generation for NLP Tasks.pdf'}, page_content='Broader Impact\nThis work offers
several positive societal bene ts over previous work: the fact that it is
more\nstrongly grounded in real factual knowledge (in this case Wikipedia) makes
it "hallucinate" less\nwith generations that are more factual, and offers more
control and interpretability. RAG could be\nemployed in a wide variety of
scenarios with direct bene t to society, for example by endowing it\nwith a
medical index and asking it open-domain questions on that topic, or by helping
people be more\neffective at their jobs.\nWith these advantages also come
potential downsides: Wikipedia, or any potential external knowledge\nsource,
will probably never be entirely factual and completely devoid of bias. Since RAG
can be\nemployed as a language model, similar concerns as for GPT-2 [50] are
valid here, although arguably\nto a lesser extent, including that it might be
used to generate abuse, faked or misleading content in')]
```

[35]:
```python
print(vector_db.similarity_search("What is RAG?", k=3)[0].page_content)
```

```
documents for questions that are less likely to bene t from retrieval,
suggesting that null document
mechanisms may not be necessary for RAG.
G Parameters
Our RAG models contain the trainable parameters for the BERT-base query and
document encoder of
DPR, with 110M parameters each (although we do not train the document encoder
ourselves) and
406M trainable parameters from BART-large, 406M parameters, making a total of
626M trainable
18
```

[36]:
```python
print(vector_db.similarity_search("What is RAG?", k=3)[1].page_content)
```

```
cases for NQ, where an extractive model would score 0%.
4.2 Abstractive Question Answering
As shown in Table 2, RAG-Sequence outperforms BART on Open MS-MARCO NLG by 2.6
Bleu
points and 2.6 Rouge-L points. RAG approaches state-of-the-art model
```

```
performance, which is
impressive given that (i) those models access gold passages with speci c
information required to
generate the reference answer , (ii) many questions are unanswerable without the
gold passages, and
(iii) not all questions are answerable from Wikipedia alone. Table 3 shows some
generated answers
from our models. Qualitatively, we  nd that RAG models hallucinate less and
generate factually
correct text more often than BART. Later, we also show that RAG generations are
more diverse than
BART generations (see §4.5).
4.3 Jeopardy Question Generation
Table 2 shows that RAG-Token performs better than RAG-Sequence on Jeopardy
question generation,
```

[37]:
```python
print(vector_db.similarity_search("What is RAG?", k=3)[2].page_content)
```

```
Broader Impact
This work offers several positive societal bene ts over previous work: the fact
that it is more
strongly grounded in real factual knowledge (in this case Wikipedia) makes it
"hallucinate" less
with generations that are more factual, and offers more control and
interpretability. RAG could be
employed in a wide variety of scenarios with direct bene t to society, for
example by endowing it
with a medical index and asking it open-domain questions on that topic, or by
helping people be more
effective at their jobs.
With these advantages also come potential downsides: Wikipedia, or any potential
external knowledge
source, will probably never be entirely factual and completely devoid of bias.
Since RAG can be
employed as a language model, similar concerns as for GPT-2 [50] are valid here,
although arguably
to a lesser extent, including that it might be used to generate abuse, faked or
misleading content in
```

[38]:
```python
from langchain.prompts import ChatPromptTemplate
Template = """
You are an assistant for question-answering tasks. Use the following pieces of␣
↪retrieved context to answer the question. If you don't know the answer, just␣
↪say that you don't know. Use three sentences maximum and keep the answer␣
↪concise.

Question: {question}
Context: {context}
```

```
Helpful Answer:

"""
```

[39]:
```python
prompt = ChatPromptTemplate.from_template(Template)
```

[40]:
```python
prompt
```

[40]:
```
ChatPromptTemplate(input_variables=['context', 'question'], input_types={},
partial_variables={}, messages=[HumanMessagePromptTemplate(prompt=PromptTemplate
(input_variables=['context', 'question'], input_types={}, partial_variables={},
template="\nYou are an assistant for question-answering tasks. Use the following
pieces of retrieved context to answer the question. If you don't know the
answer, just say that you don't know. Use three sentences maximum and keep the
answer concise.\n\nQuestion: {question}\nContext: {context}\nHelpful
Answer:\n\n"), additional_kwargs={})])
```

[47]:
```python
from google.colab import userdata
api_token = userdata.get('huggingface_api_token')
```

[1]:
```python
api_token #"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"
```

[49]:
```python
from langchain import HuggingFaceHub
model = HuggingFaceHub(
     huggingfacehub_api_token = api_token,
     repo_id="mistralai/Mistral-7B-Instruct-v0.1",
                        model_kwargs={"temperature":1,
                                      "max_length":128})
```

[50]:
```python
from langchain.schema import StrOutputParser
from langchain.schema.runnable import RunnablePassthrough
```

[51]:
```python
output_parser = StrOutputParser()
retriever = vector_db.as_retriever()
rag_chain = (
    {"context": retriever, "question": RunnablePassthrough()}
    | prompt
    | model
    | output_parser
)
```

[54]:
```python
#rag_chain.invoke("What is rag system?") # Results were very good but due to
 ↪free plan and many request this is error
```

[ ]:
```python

```