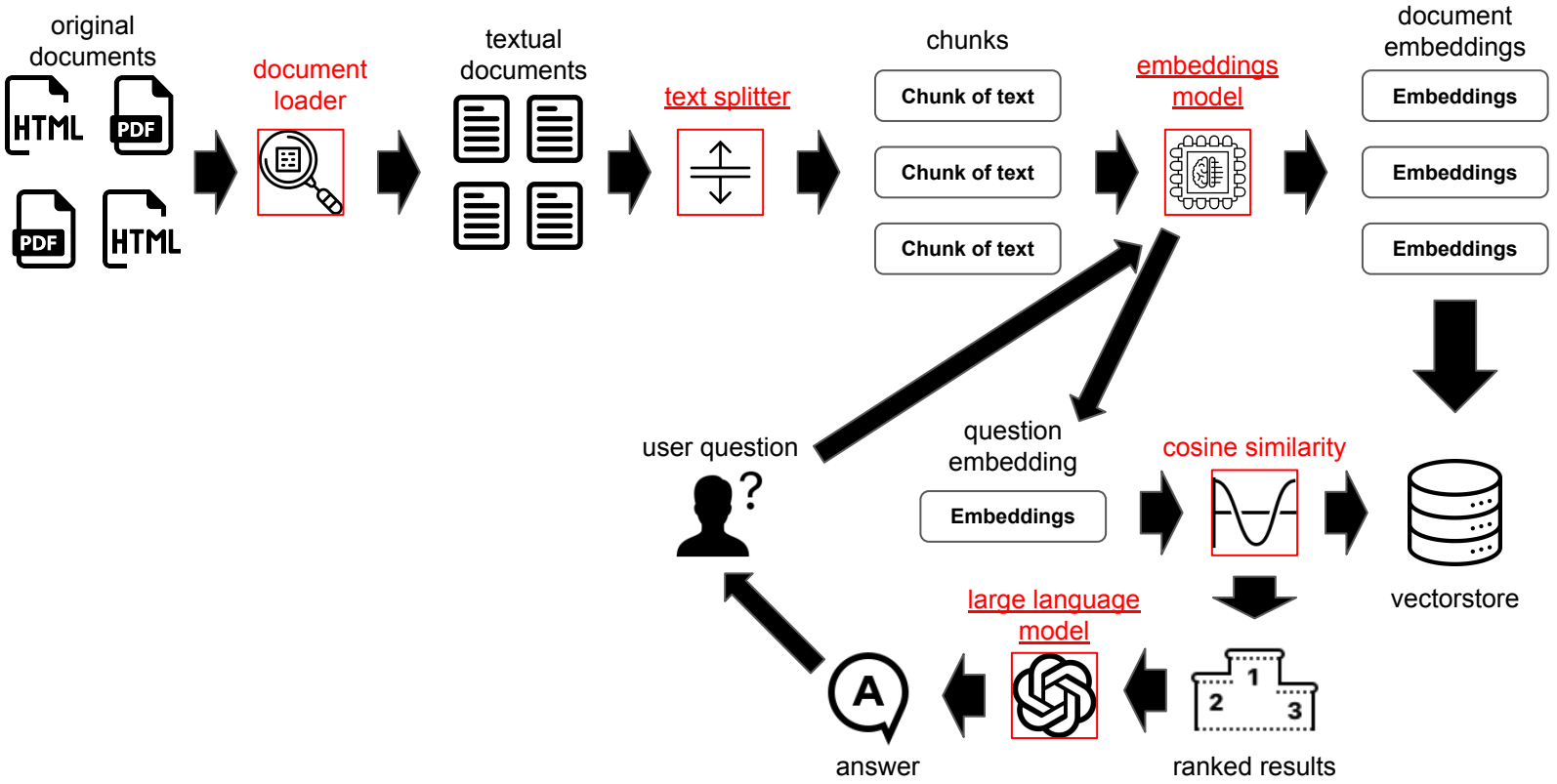
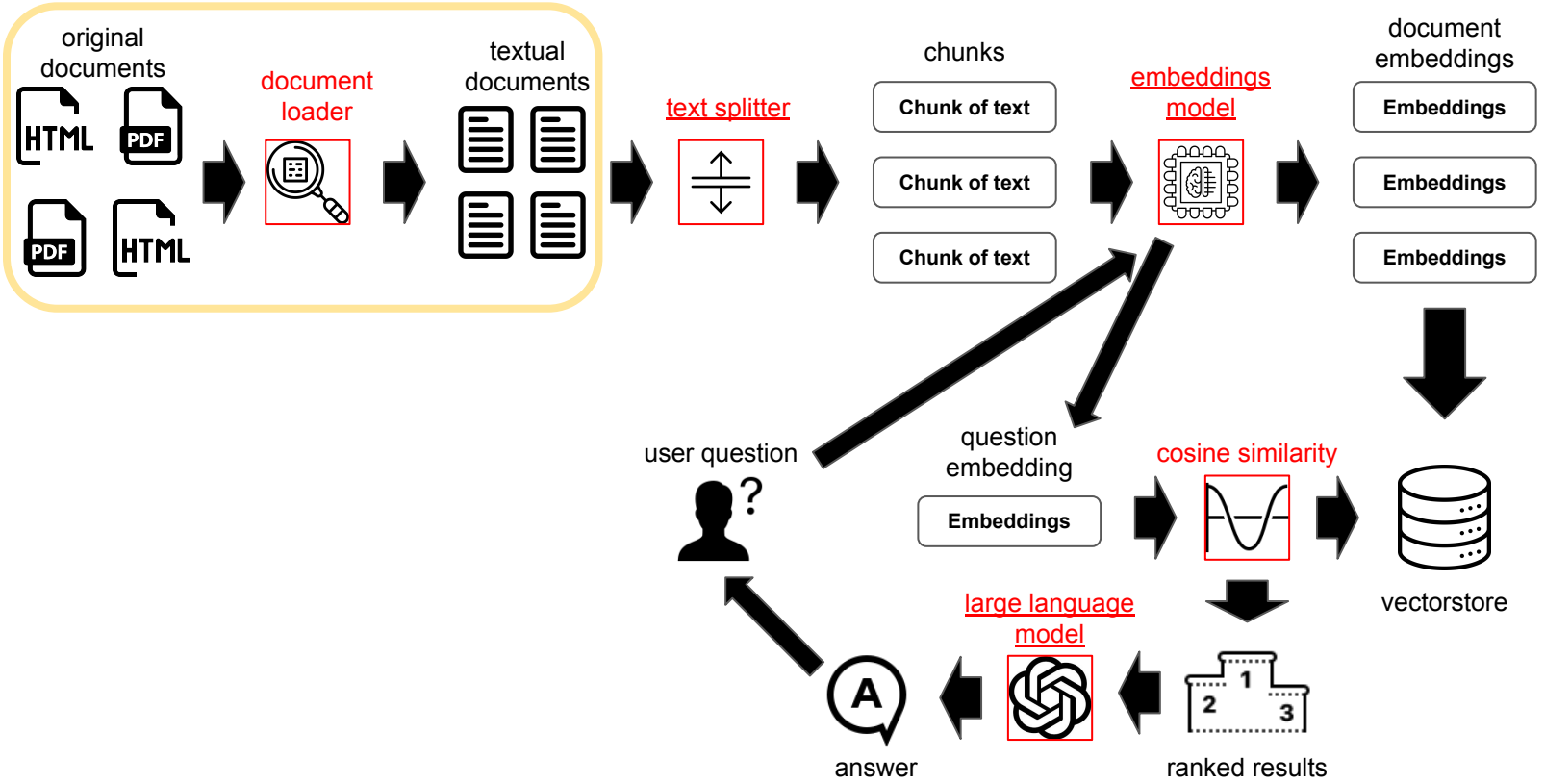
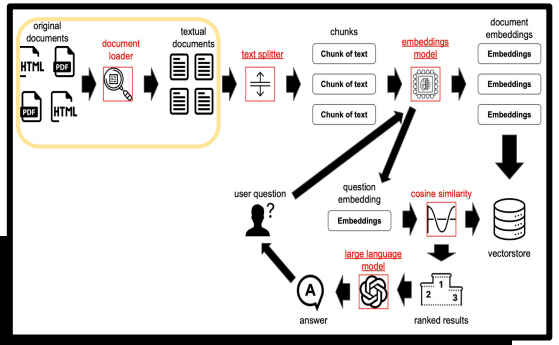
Langchain


original documents
HTML
PDF
PDF
HTML
document loader
textual documents
text splitter
chunks
Chunk of text
Chunk of text
Chunk of text
embeddings model
document embeddings
Embeddings
Embeddings
Embeddings
vectorstore
user question
question embedding
Embeddings
cosine similarity
large language model
A
answer
1
2
3
ranked results

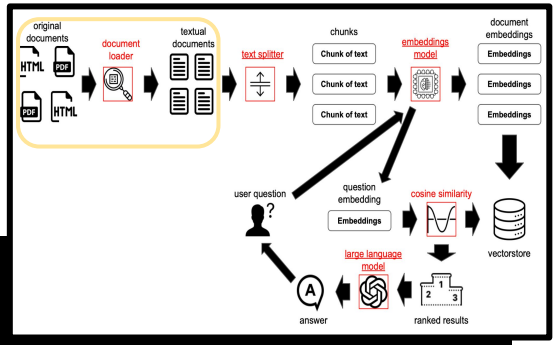

original documents
HTML
PDF
PDF
HTML
document loader
textual documents
text splitter
chunks
Chunk of text
Chunk of text
Chunk of text
embeddings model
document embeddings
Embeddings
Embeddings
Embeddings
vectorstore
user question
question embedding
Embeddings
cosine similarity
large language model
A
answer
1
2
3
ranked results

Document loader (PDF)



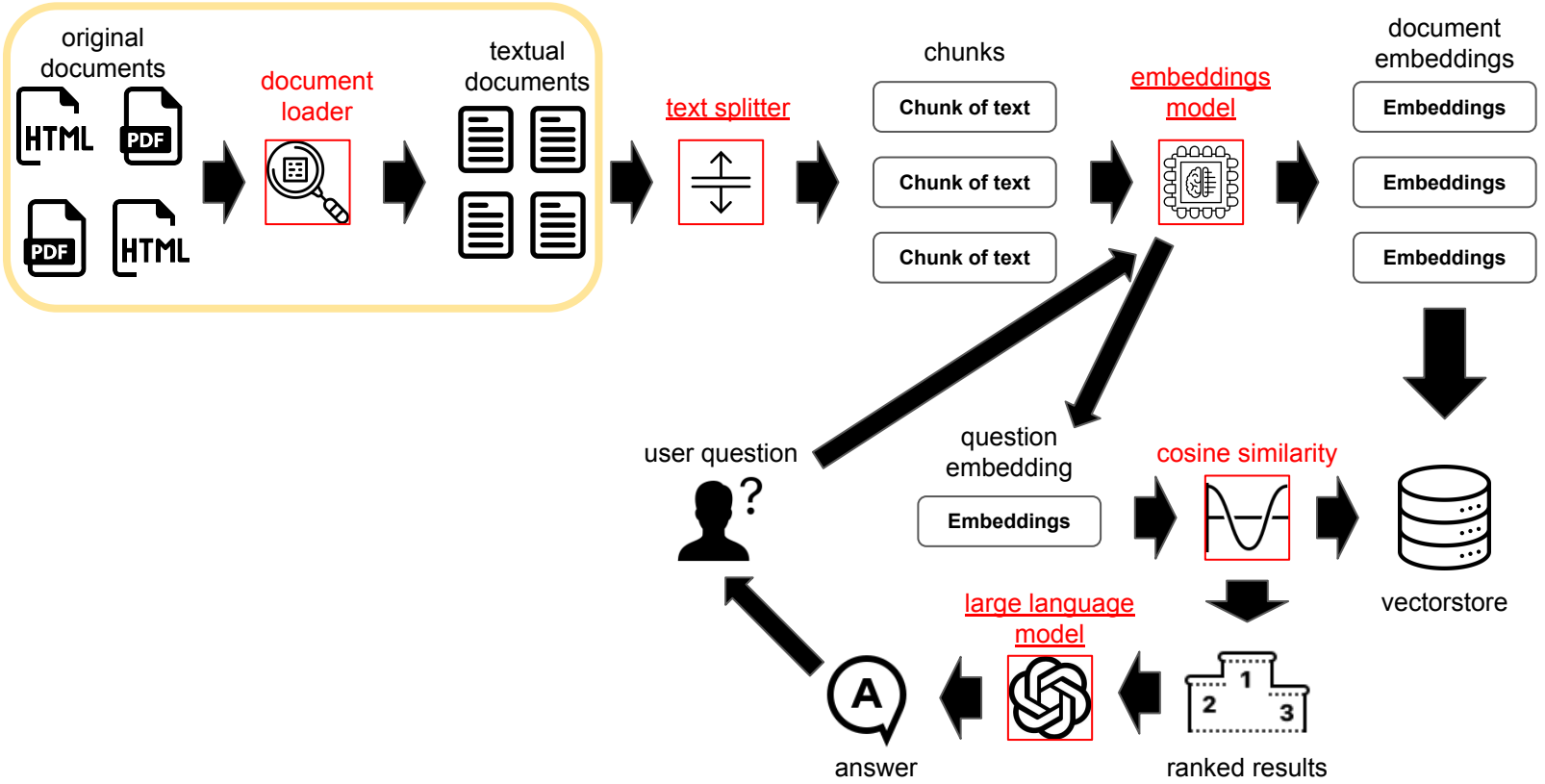

```python
#Importazione delle librerie PyPDFLoader e os
from langchain.document_loaders import PyPDFLoader
import os
# Definisci il percorso alla cartella contenente i file PDF da elaborare
path_to_PDF_files = "files_for_chatbot/PDF/"
# Ottieni la lista di tutti i file nella cartella specificata
files = os.listdir(path_to_PDF_files)
# Crea una lista vuota chiamata 'documents' per contenere i documenti estratti dai file PDF
documents = list()
# Loop attraverso ogni file nella lista 'files'
for f in files:
    # Crea un oggetto 'loader' per PyPDFLoader specifico per il file PDF corrente
    loader = PyPDFLoader(path_to_PDF_files + f)
    # Carica e suddivide il contenuto del file PDF corrente
    documents += loader.load_and_split()
```

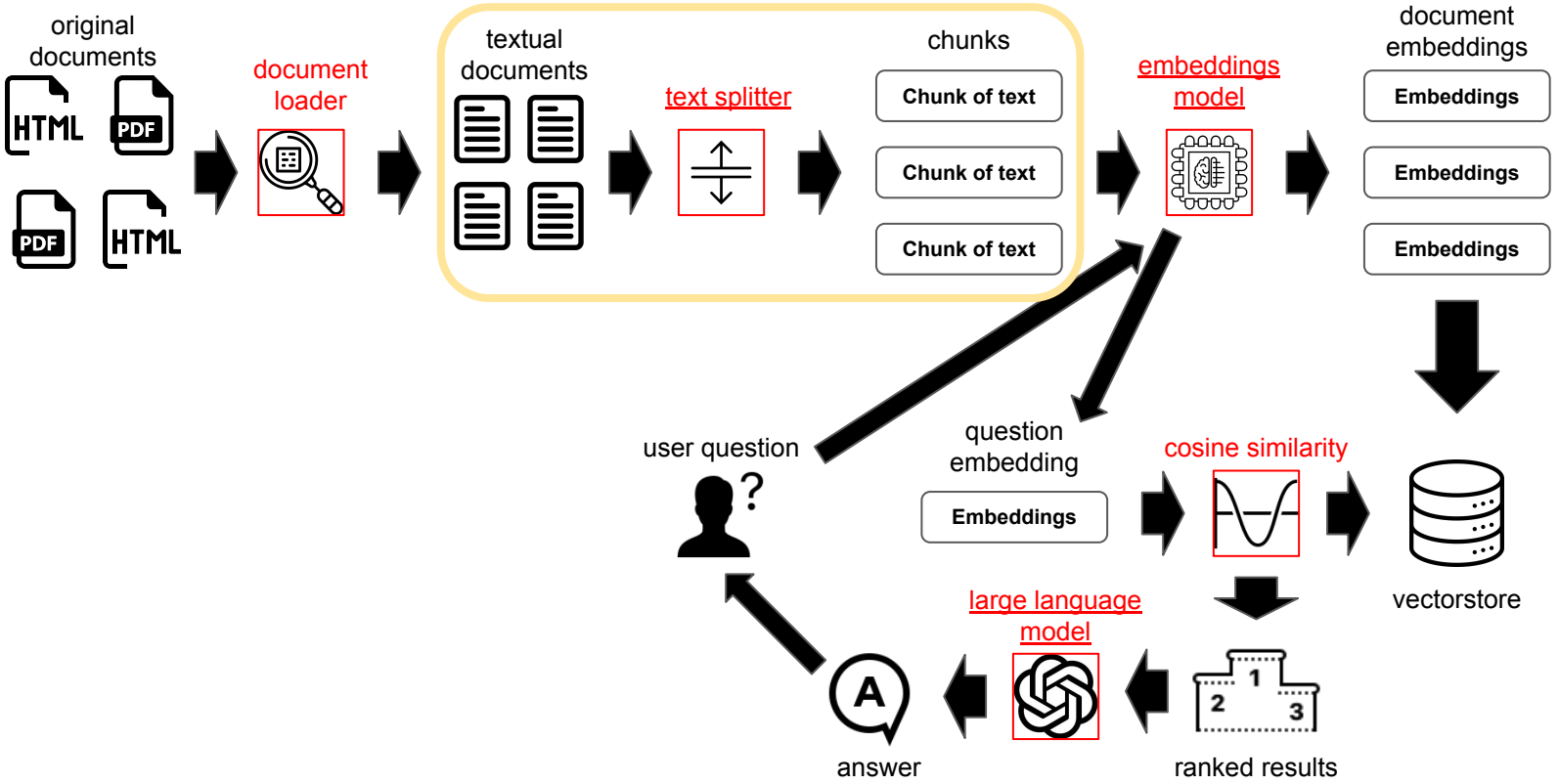
Document loader (HTML)



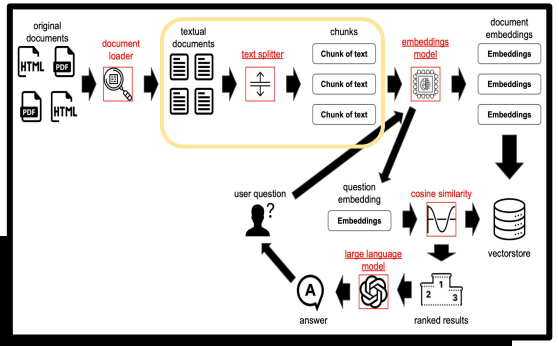

```python
from langchain.document_loaders  import UnstructuredURLLoader
# Definisci una lista di URL da cui scaricare contenuto
urls = [
    "https://ourworldindata.org/artificial-intelligence" ,
    "https://ourworldindata.org/causes-of-death" ,
    "https://ourworldindata.org/terrorism"
]
# Crea un oggetto 'loader' di tipo UnstructuredURLLoader, specificando gli URL da caricare
loader = UnstructuredURLLoader (urls=urls)
# Carica il contenuto dalle URL specificate
documents = loader.load ()
```

Altri Loaders: https://python.langchain.com/docs/modules/data_connection/document_loaders/


original documents
HTML
PDF
PDF
HTML
document loader
textual documents
text splitter
chunks
Chunk of text
Chunk of text
Chunk of text
embeddings model
document embeddings
Embeddings
Embeddings
Embeddings
vectorstore
user question
question embedding
Embeddings
cosine similarity
large language model
A
answer
1
2
3
ranked results

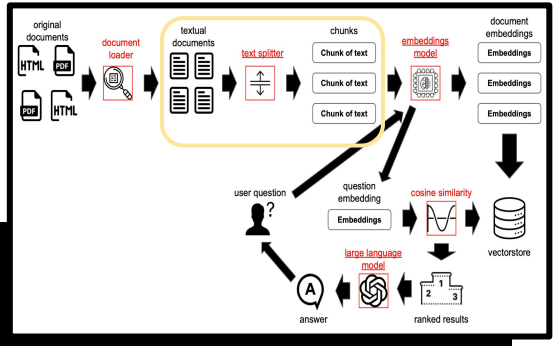

original documents
HTML
PDF
PDF
HTML
document loader
textual documents
text splitter
chunks
Chunk of text
Chunk of text
Chunk of text
embeddings model
document embeddings
Embeddings
Embeddings
Embeddings
vectorstore
user question
question embedding
Embeddings
cosine similarity
large language model
A
answer
1
2
3
ranked results

Text splitter



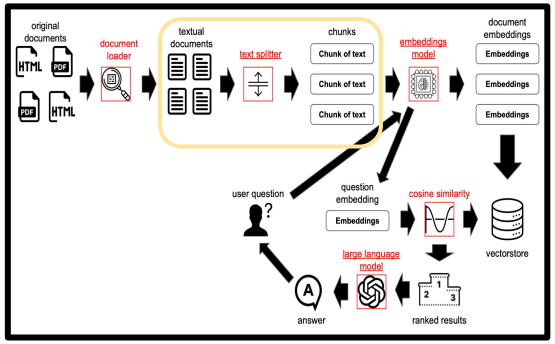

```python
import nltk
nltk.download('punkt')
def split_text_into_groups(text, n=3, overlap=2, lang="italian"):
    # Tokenizza il testo in frasi sulla base della lingua specificata
    sentences = nltk.sent_tokenize(text, language=lang)
    # Inizializza una lista per i gruppi di frasi
    groups = []
    # Suddivide il testo in gruppi
    for i in range(0, len(sentences), n - overlap):
        if i + n < len(sentences):
            group = ' '.join(sentences[i:i + n])
            groups.append(group)
    # Restituisce i gruppi di frasi
    return groups
```

Text splitter



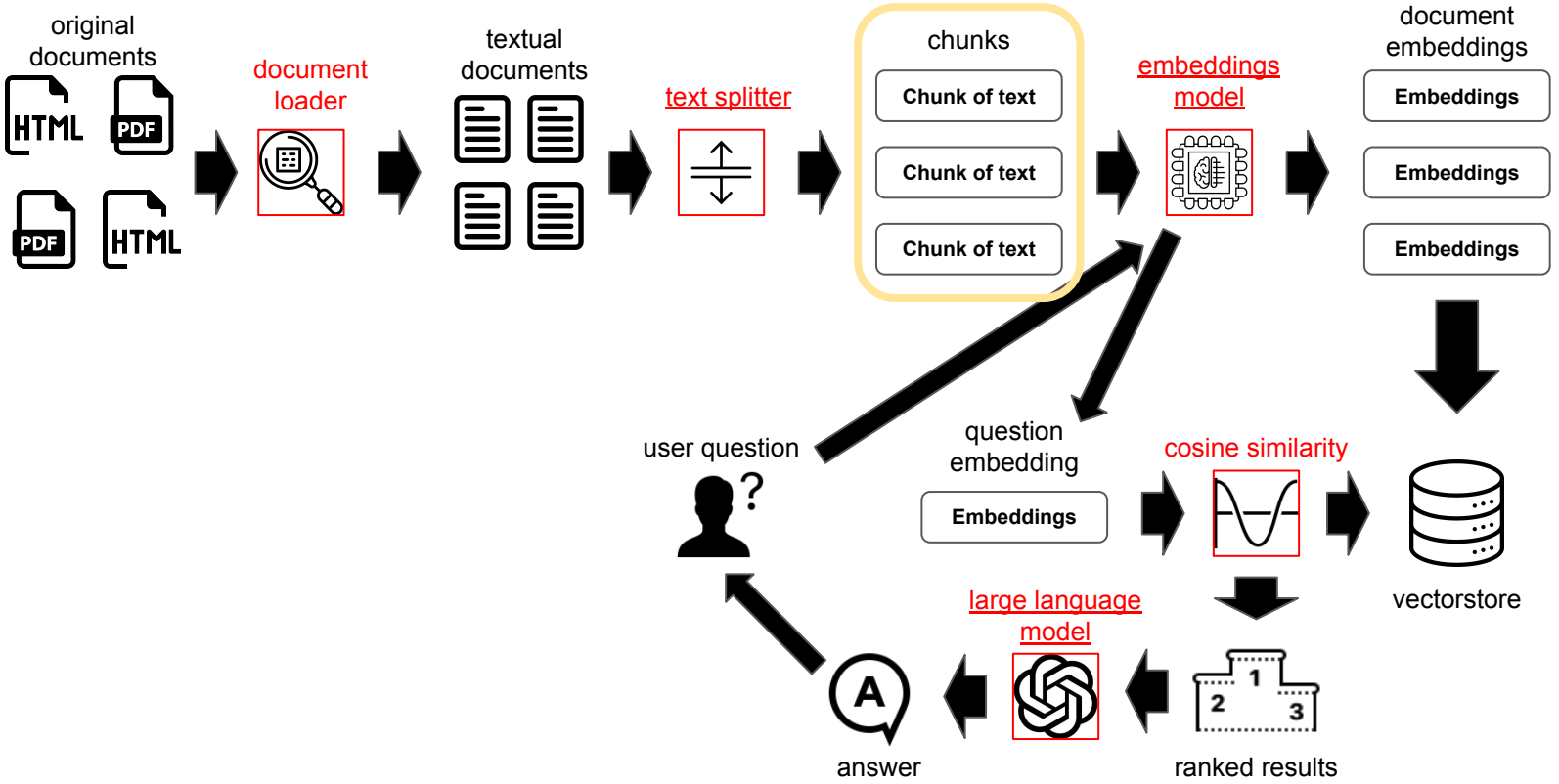

```python
from langchain.docstore.document import Document
# Lista vuota per memorizzare i documenti divisi
documents_splitted = list()
# Contatore per generare un ID univoco per ciascun documento
document_id = 0
# Itera sui primi 20 documenti per velocizzare l'elaborazione
for original_doc in documents:
    # Split del contenuto del documento in gruppi
    for group_content in split_text_into_groups(original_doc.page_content):
        # Copia dei metadati originali del documento
        metadata = original_doc.metadata
        # Assegna un ID univoco al documento
        metadata['id'] = document_id
        # Incremento dell'ID per il prossimo documento
        document_id += 1
        # Creazione di un nuovo documento con il contenuto diviso e i metadati aggiornati
        doc_splitted = Document(page_content=group_content, metadata=metadata)
        # Aggiungi il documento diviso alla lista
        documents_splitted.append(doc_splitted)
```

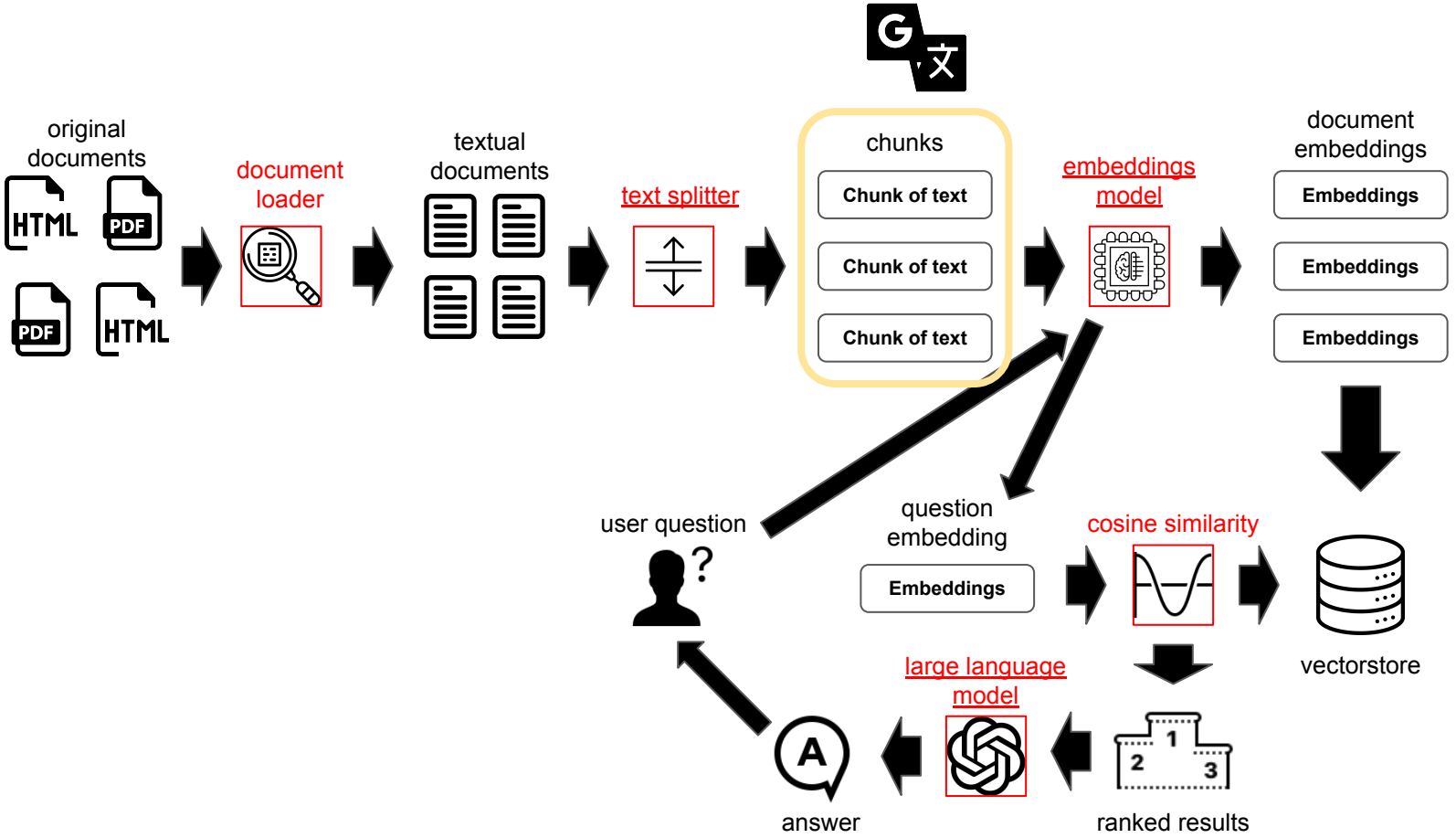
Text splitter



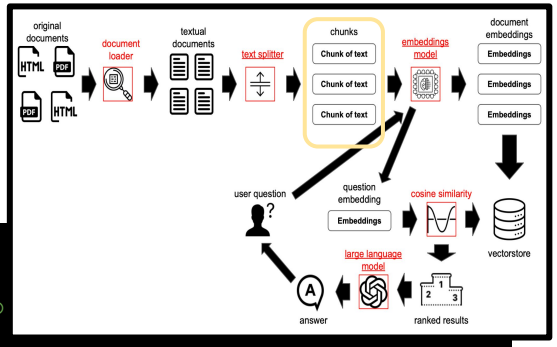
Altri splitter:

https://python.langchain.com/docs/modules/data_connection/document_transformers/#


original documents
HTML
PDF
PDF
HTML
document loader
textual documents
text splitter
chunks
Chunk of text
Chunk of text
Chunk of text
embeddings model
document embeddings
Embeddings
Embeddings
Embeddings
vectorstore
user question
question embedding
Embeddings
cosine similarity
large language model
A
answer
1
2
3
ranked results

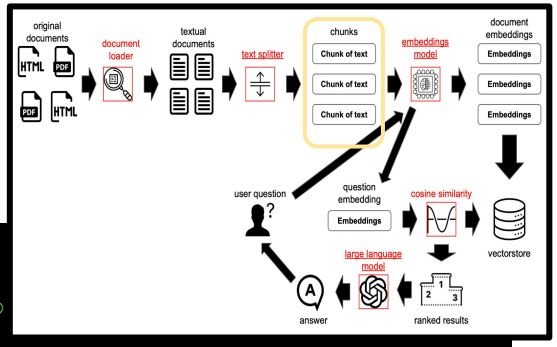

original documents
HTML
PDF
PDF
HTML
document loader
textual documents
text splitter
chunks
Chunk of text
Chunk of text
Chunk of text
embeddings model
document embeddings
Embeddings
Embeddings
Embeddings
vectorstore
user question
question embedding
Embeddings
cosine similarity
large language model
A
answer
1
2
3
ranked results

Italian > English

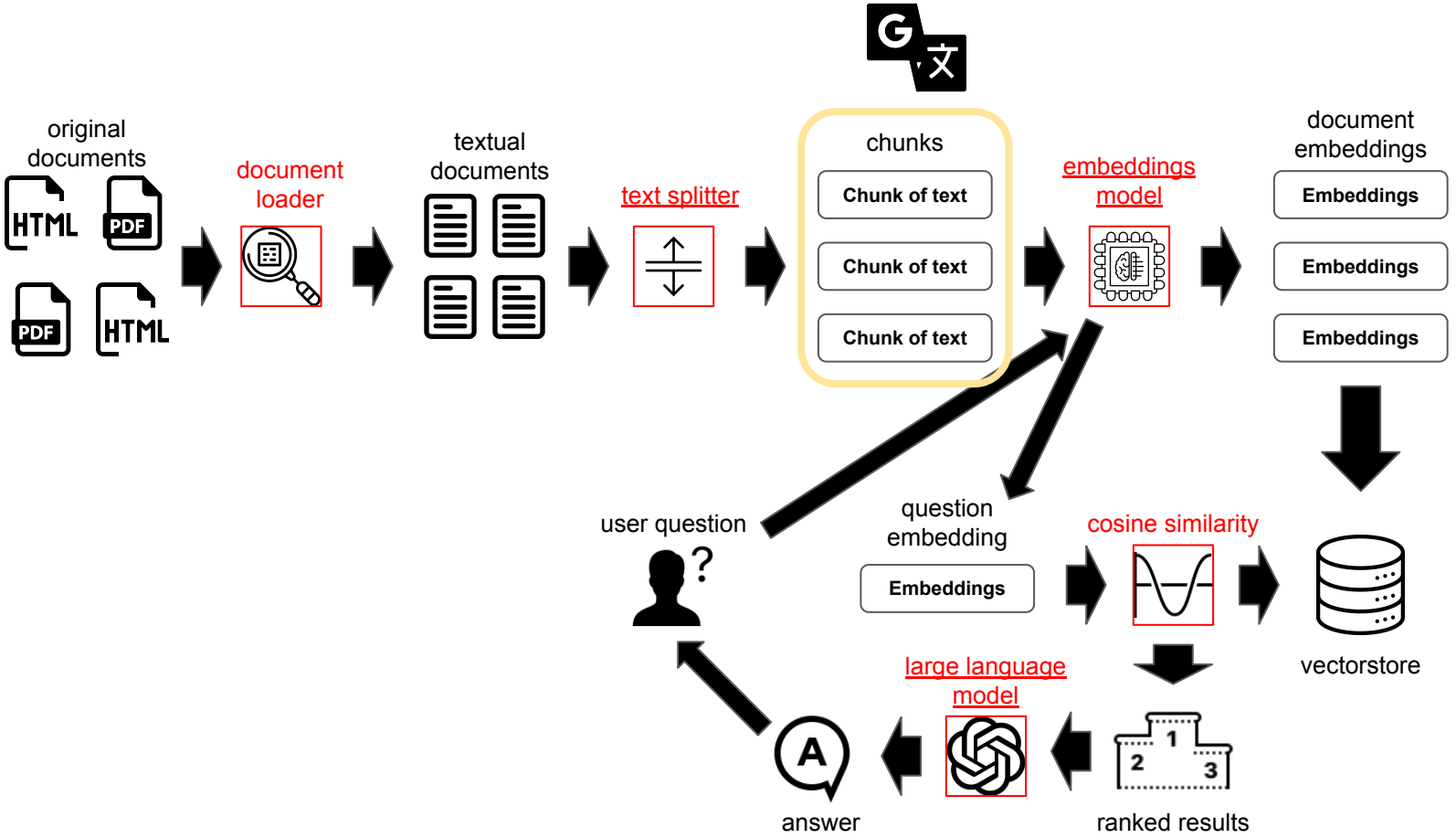


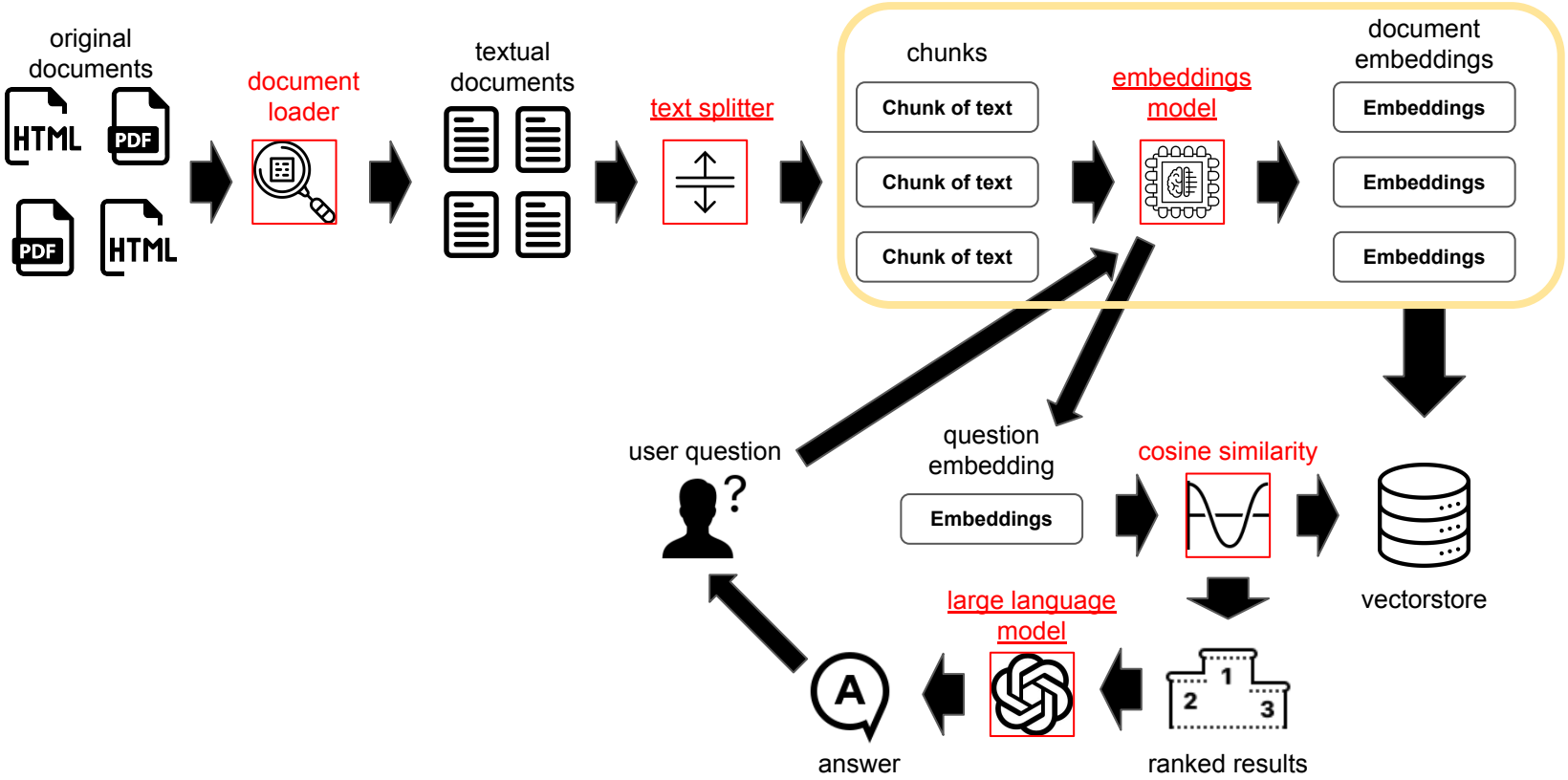

```
from googletrans import Translator
# Definizione di una funzione che utilizza googletrans per tradurre il testo
def google_translator(text, from_code="it", to_code="en"):
    translator = Translator()
    # Traduzione del testo dalla lingua di origine alla lingua di destinazione
    translation = translator.translate(text, dest=to_code, src=from_code).text
    return translation
# Lista vuota per memorizzare i documenti tradotti in inglese
documents_splitted_en = list()
# Iterazione su una lista di documenti suddivisi
for doc in documents_splitted:
    # Traduzione del contenuto della pagina del documento corrente dall'italiano all'inglese
    doc_splitted_en = Document(
        page_content=google_translator(doc.page_content),
        metadata=doc.metadata
    )
    # Aggiunta del documento tradotto (in inglese) alla lista
    documents_splitted_en.append(doc_splitted_en)
```

Italian > English

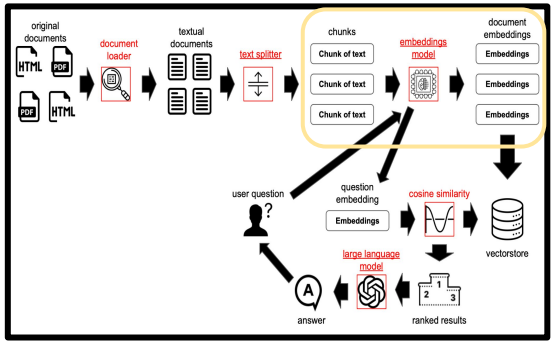



```
from googletrans import Translator
# Definizione di una funzione che utilizza googletrans per tradurre il testo
def google_translator(text, from_code="it", to_code="en"):
    translator = Translator()
    # Traduzione del testo dalla lingua di origine alla lingua di destinazione
    translation = translator.translate(text, dest=to_code, src=from_code).text
    return translation
# Lista vuota per memorizzare i documenti tradotti in inglese
documents_splitted_en = list()
# Iterazione su una lista di documenti suddivisi
for doc in documents_splitted:
    # Traduzione del contenuto della pagina del documento corrente dall'italiano all'inglese
    doc_splitted_en = Document(
        page_content=google_translator(doc.page_content),
        metadata=doc.metadata
    )
    # Aggiunta del documento tradotto (in inglese) alla lista
    documents_splitted_en.append(doc_splitted_en)
```


original documents
HTML
PDF
PDF
HTML
document loader
textual documents
text splitter
chunks
Chunk of text
Chunk of text
Chunk of text
embeddings model
document embeddings
Embeddings
Embeddings
Embeddings
vectorstore
user question
question embedding
Embeddings
cosine similarity
large language model
A
answer
1
2
3
ranked results


original documents
HTML
PDF
PDF
HTML
document loader
textual documents
text splitter
chunks
Chunk of text
Chunk of text
Chunk of text
embeddings model
document embeddings
Embeddings
Embeddings
Embeddings
vectorstore
user question
question embedding
Embeddings
cosine similarity
large language model
A
answer
1
2
3
ranked results

Che cos’è un embedding?



Supponiamo di avere le seguenti tre frasi e i relativi vettori di embedding:

- "La musica rende felici le persone."

Vettore di embedding: `[0.5, 0.7, -0.3, 0.1]`

- "I brani musicali portano gioia alle persone."

Vettore di embedding: `[0.6, 0.8, -0.2, 0.2]`

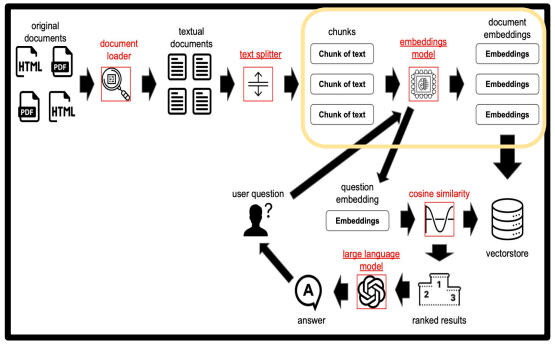
- "Le montagne sono alte e coperte di neve."

Vettore di embedding: `[0.4, -0.6, 0.9, -0.5]`

Nel primo esempio, le frasi "La musica rende felici le persone." e "I brani musicali portano gioia alle persone." sono semanticamente simili, e ciò è riflesso nella somiglianza tra i loro vettori di embedding.

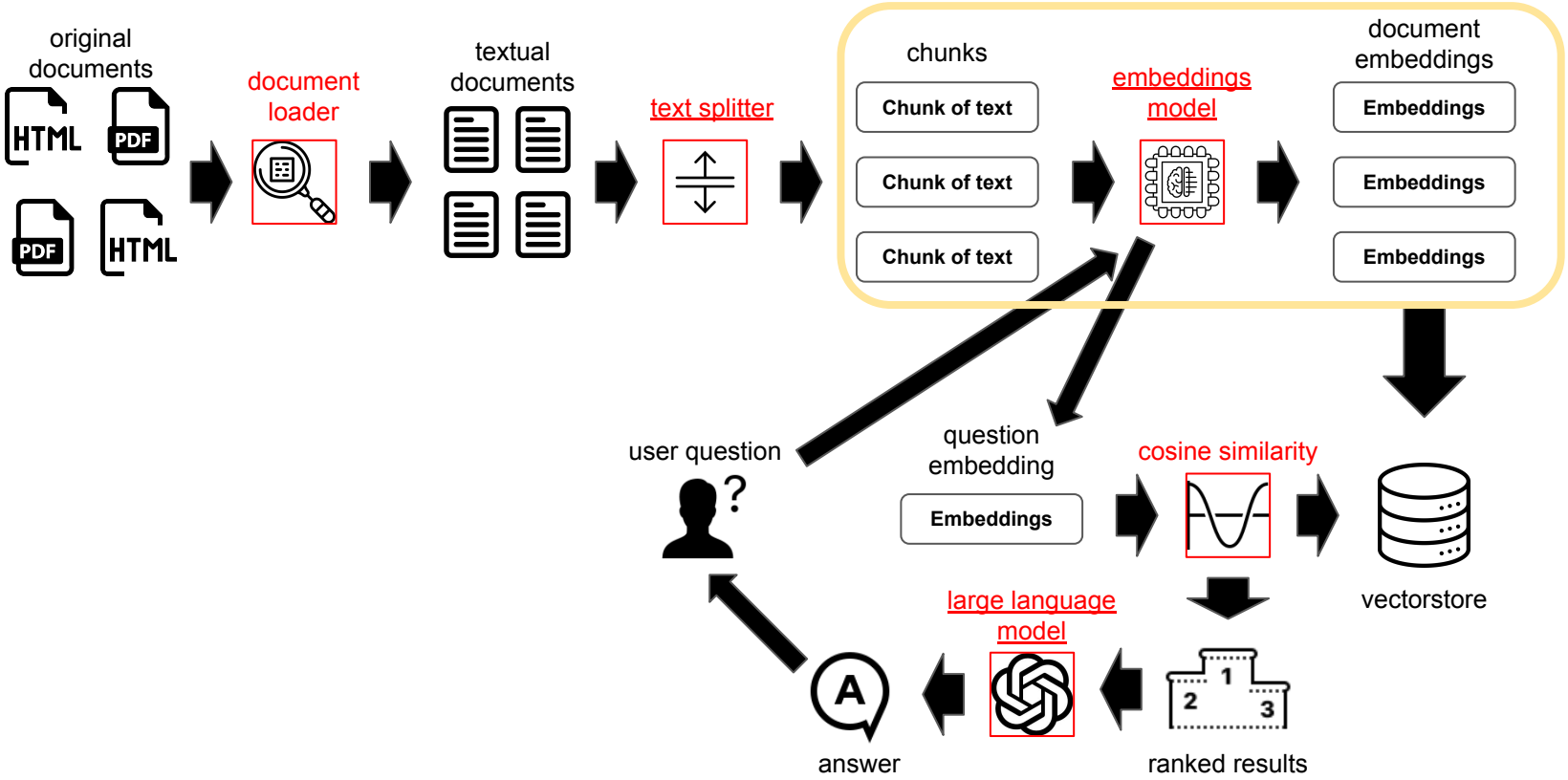
Nel terzo esempio, la frase "Le montagne sono alte e coperte di neve." è diversa dalle prime due. La rappresentazione vettoriale per questa frase sarà significativamente diversa, riflettendo la differente semantica rispetto alle frasi precedenti.

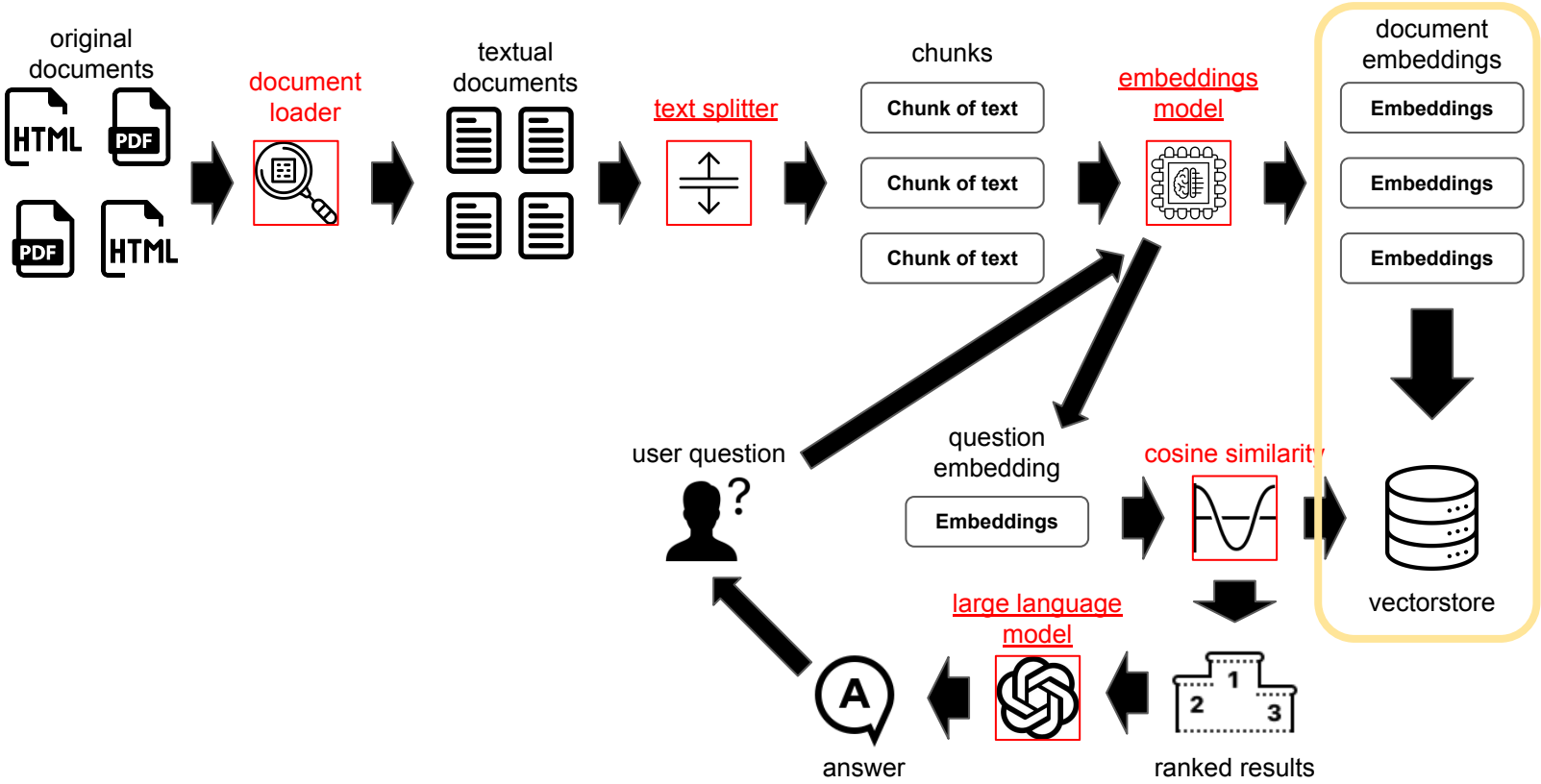
Sentence Transformer



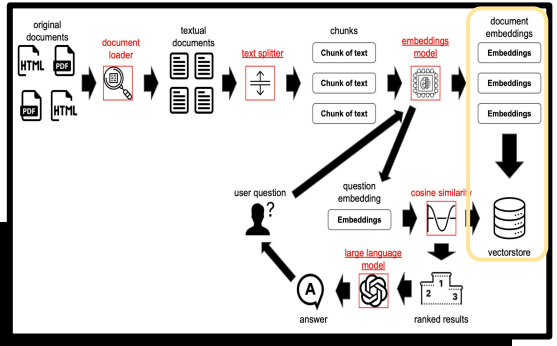
Scegliere il modello di Sentence Transformer per gli embedding:
https://huggingface.co/models?library=sentence-transformers

```
from langchain.embeddings import HuggingFaceEmbeddings
embeddings_model_name = "all-mpnet-base-v2"
embedding_model = HuggingFaceEmbeddings(
    model_name=embeddings_model_name,
    model_kwargs={"device": "cpu"},
    encode_kwargs={'normalize_embeddings': True}
)
```

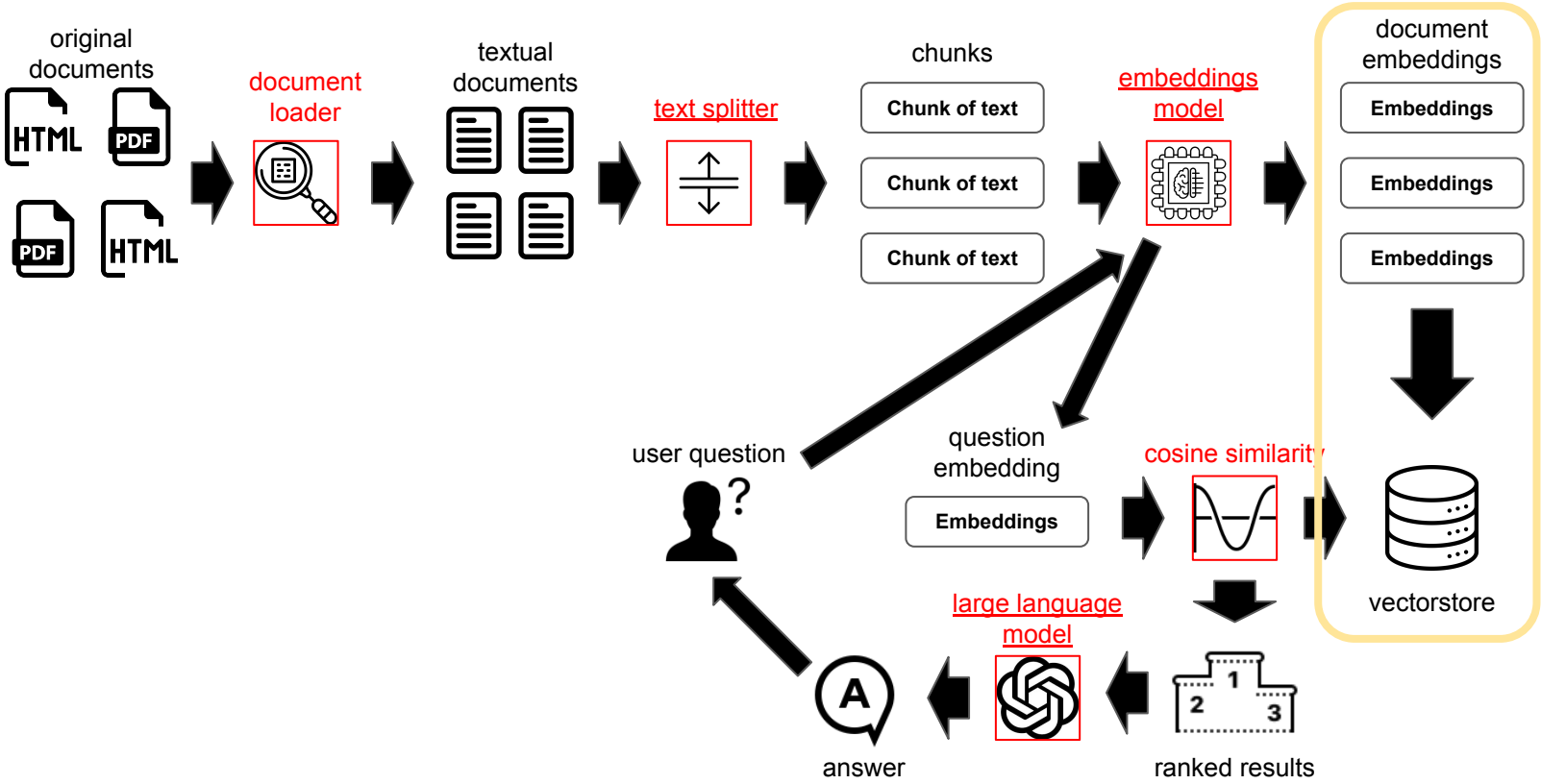

original documents
HTML
PDF
PDF
HTML
document loader
textual documents
text splitter
chunks
Chunk of text
Chunk of text
Chunk of text
embeddings model
document embeddings
Embeddings
Embeddings
Embeddings
vectorstore
user question
question embedding
Embeddings
cosine similarity
large language model
A
answer
1
2
3
ranked results

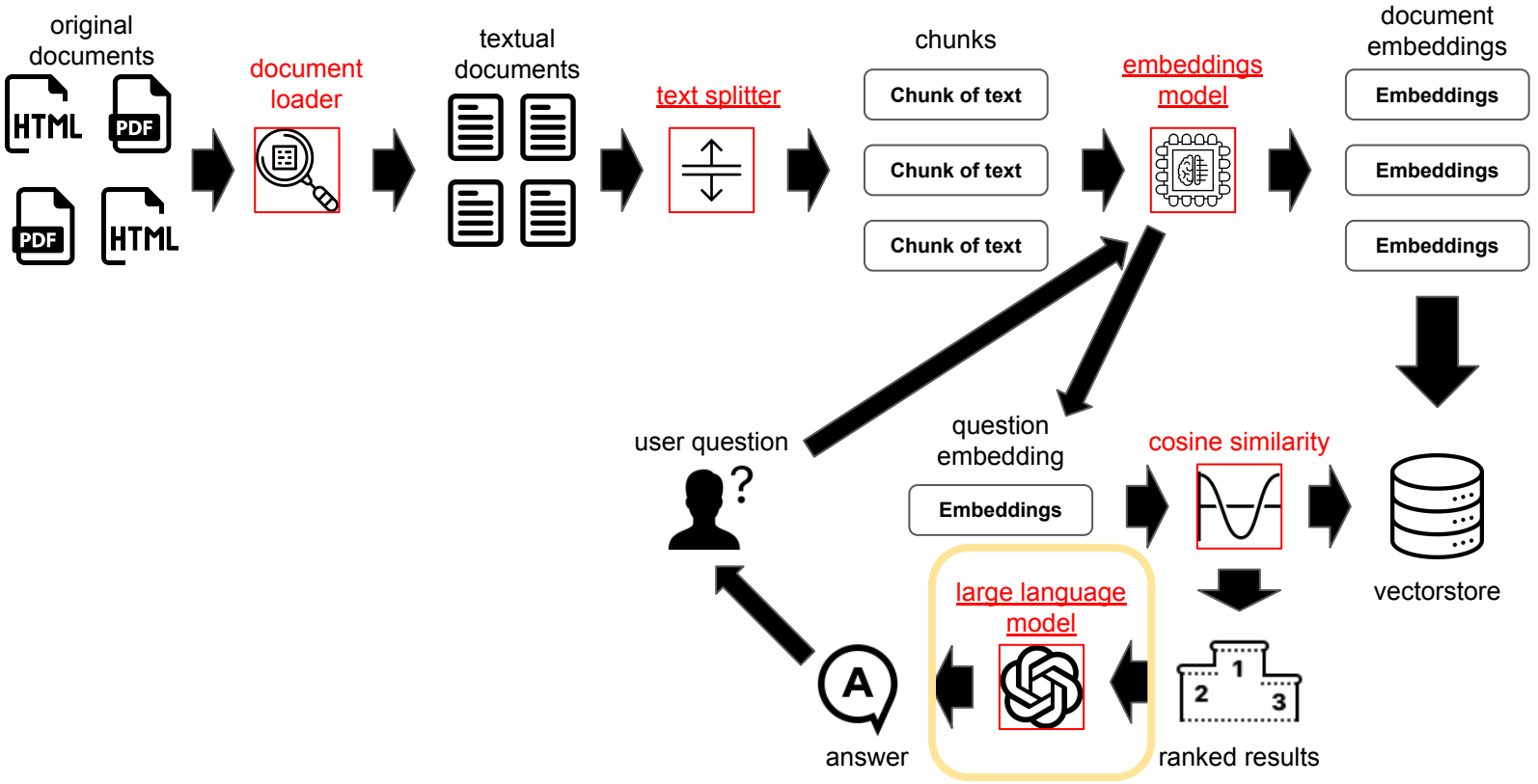

original documents
HTML
PDF
PDF
HTML
document loader
textual documents
text splitter
chunks
Chunk of text
Chunk of text
Chunk of text
embeddings model
document embeddings
Embeddings
Embeddings
Embeddings
vectorstore
user question
question embedding
Embeddings
cosine similarity
large language model
A
answer
1
2
3
ranked results

Vectorstore

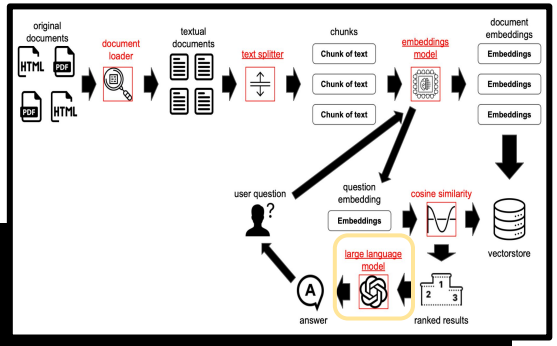



```
from langchain.vectorstores import Chroma
db = Chroma.from_documents(
    documents_splitted_en,
    embedding_model,
    collection_metadata={"hnsw:space": "cosine"}
)
```

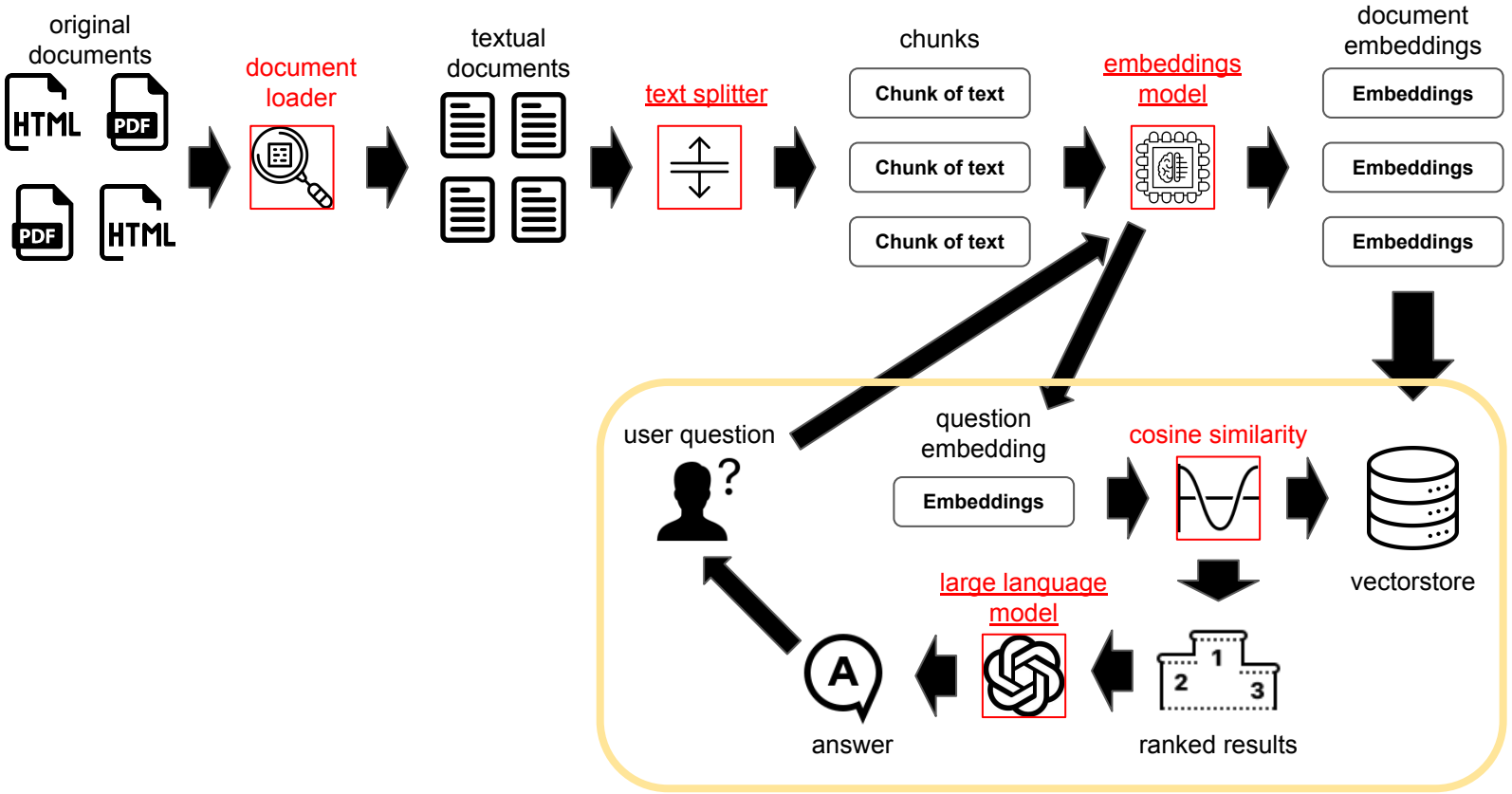

original documents
HTML
PDF
PDF
HTML
document loader
textual documents
text splitter
chunks
Chunk of text
Chunk of text
Chunk of text
embeddings model
document embeddings
Embeddings
Embeddings
Embeddings
vectorstore
user question
question embedding
Embeddings
cosine similarity
large language model
A
answer
1
2
3
ranked results


original documents
HTML
PDF
PDF
HTML
document loader
textual documents
text splitter
chunks
Chunk of text
Chunk of text
Chunk of text
embeddings model
document embeddings
Embeddings
Embeddings
Embeddings
vectorstore
user question
question embedding
Embeddings
cosine similarity
large language model
A
answer
1
2
3
ranked results

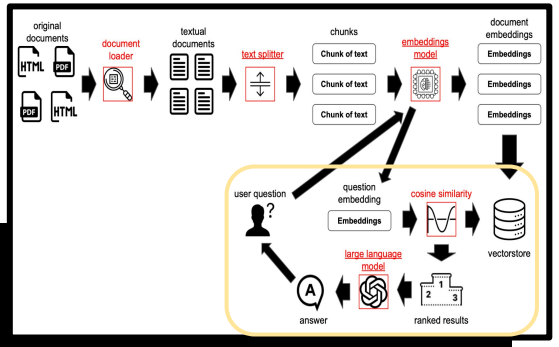
Definizione del Large Language Model



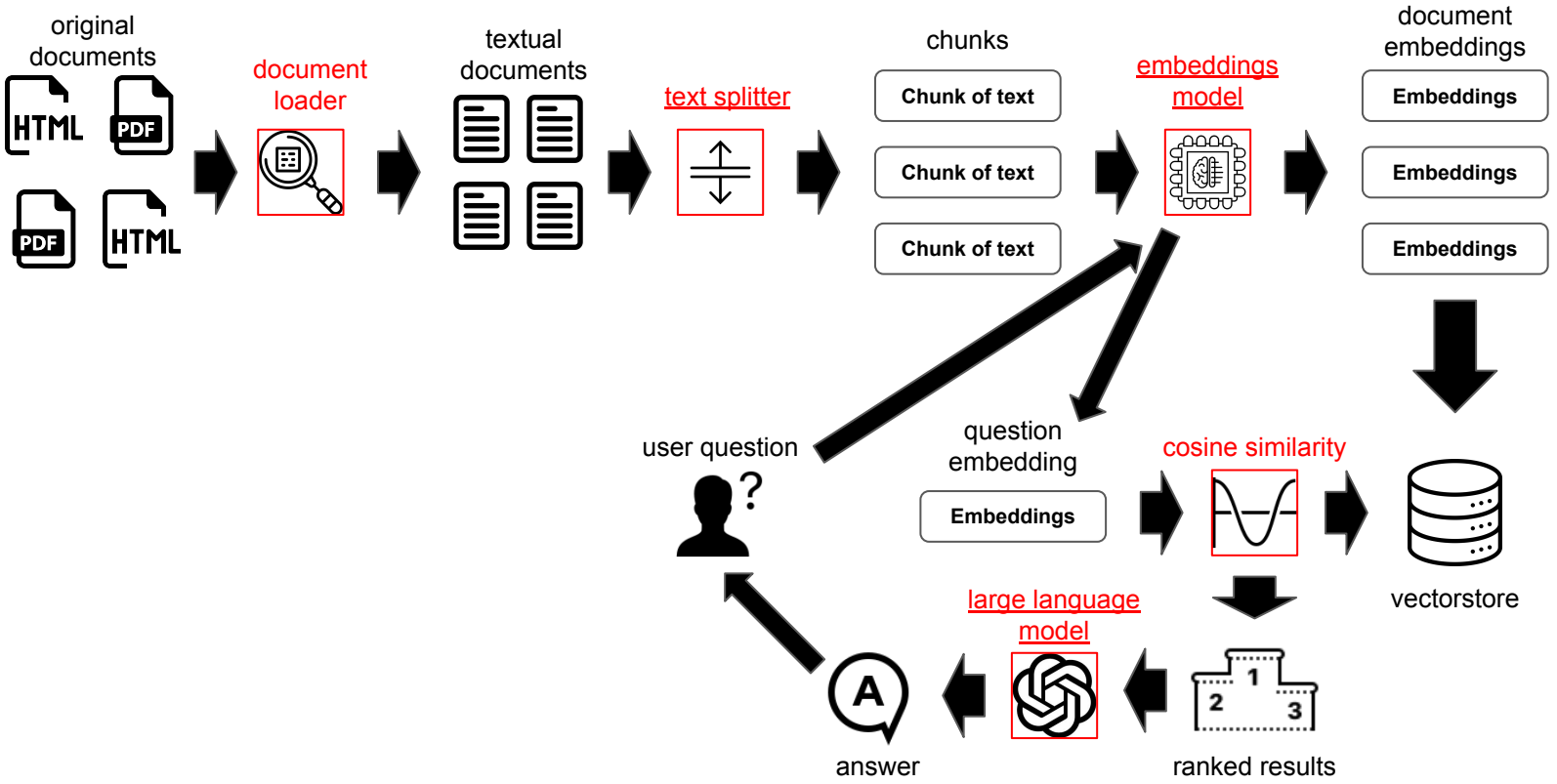

```
# Creazione di una directory per memorizzare i file del modello
!mkdir flan-alpaca-large
# Importazione della funzione snapshot_download dal modulo huggingface_hub
from huggingface_hub import snapshot_download
# Scaricamento del modello da Hugging Face Model Hub e memorizzazione nella directory locale
snapshot_download(repo_id="declare-lab/flan-alpaca-large", local_dir='./flan-alpaca-large/',
local_dir_use_symlinks=False, token="hf_...")
# Importazione della classe HuggingFacePipeline dal modulo langchain.llms
from langchain.llms import HuggingFacePipeline
# Specifica del percorso alla directory contenente i file del modello
model_dir = "./flan-alpaca-large/"
# Inizializzazione di una pipeline per l'uso del modello scaricato
llm = HuggingFacePipeline.from_model_id(model_id=model_dir,
                                        task='text2text-generation',
                                        model_kwargs={"temperature": 0.60,
                                                      "min_length": 35,"max_length": 500,
                                                     "repetition_penalty": 5.0}
)
```


original documents
HTML
PDF
PDF
HTML
document loader
textual documents
text splitter
chunks
Chunk of text
Chunk of text
Chunk of text
embeddings model
document embeddings
Embeddings
Embeddings
Embeddings
user question
question embedding
Embeddings
cosine similarity
vectorstore
large language model
A
answer
1
2
3
ranked results

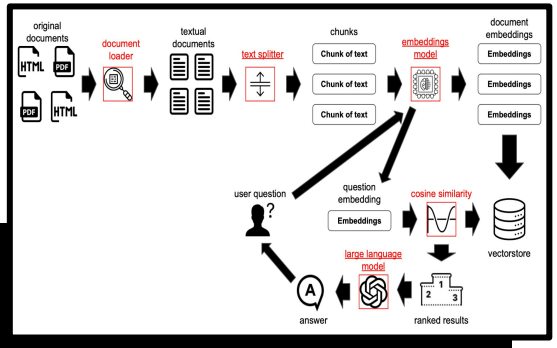
Definizione del Retriever




```python
# Importazione delle librerie e classi necessarie
from langchain.chains import RetrievalQA
# Importa la classe RetrievalQA dal modulo langchain.chains
from langchain.llms import OpenAI  # Importa la classe OpenAI dal modulo langchain.llms
# Creazione di un oggetto RetrievalQA
qa = RetrievalQA.from_chain_type(
    llm=llm,  # Specifica il modello di linguaggio da utilizzare, precedentemente inizializzato
    chain_type="stuff",  # Tipo di catena (chain) da utilizzare (può variare in base all'applicazione)
    retriever=db.as_retriever(),  # Configura il chromadb precedentemente creato come retriever
    return_source_documents=True,  # Opzione per restituire i documenti sorgente
    verbose=False  # Disattiva la modalità verbosa (output dettagliato)
)
```

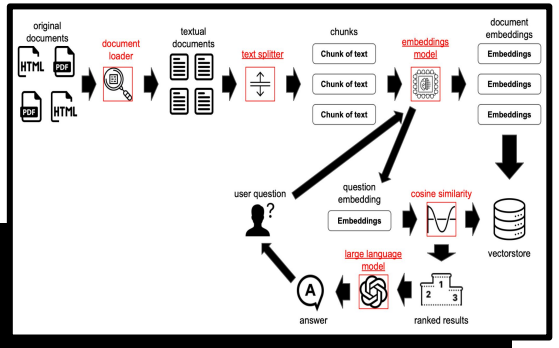

original documents
HTML
PDF
PDF
HTML
document loader
textual documents
text splitter
chunks
Chunk of text
Chunk of text
Chunk of text
embeddings model
document embeddings
Embeddings
Embeddings
Embeddings
vectorstore
user question
question embedding
Embeddings
cosine similarity
large language model
A
answer
1
2
3
ranked results

Test del Chatbot




```python
import warnings
# Funzione per ottenere una risposta dalla chat
def get_chat_response(text, qa=qa, lang='it'):
   # Disabilita i warning all'interno della funzione
   with warnings.catch_warnings():
     warnings.filterwarnings("ignore")
     if lang == "it":
         # Traduci il testo in inglese utilizzando google_translator
         text_en = google_translator(text)
         # Esegui una query usando la pipeline di RetrievalQA (qa) sul testo in inglese
         res = qa(
             text_en
         )
         # Traduci la risposta in italiano
         res["query"] = text
         res["result"] = google_translator(res["result"], from_code="en", to_code="it")
     else:
         # Esegui una query usando la pipeline di RetrievalQA (qa) sul testo in lingua originale
         res = qa(
             text
         )
   return res
```

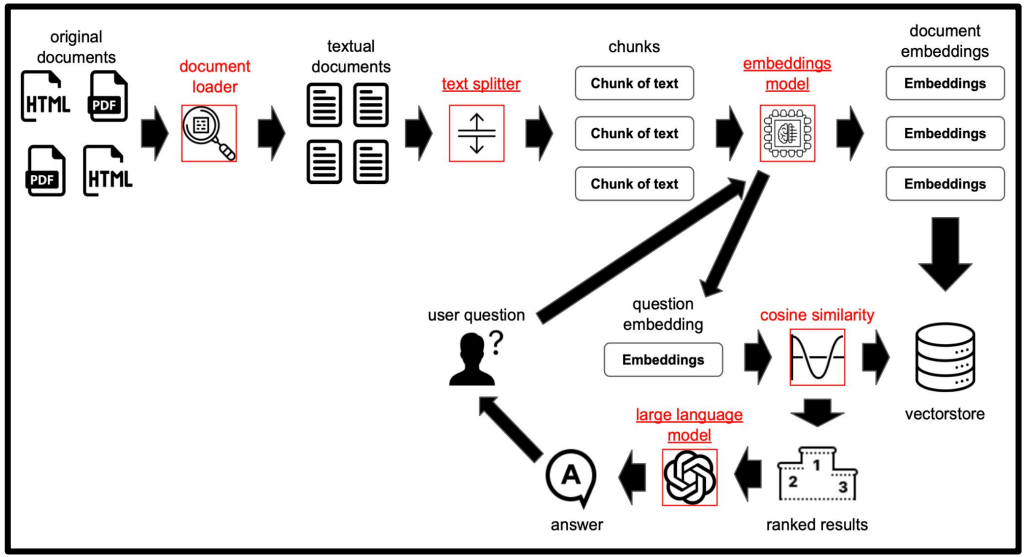
Test del Chatbot




```
# Definizione di una query
query = "Cos'è il principio di Pareto?"
print("Query:", query)

# Ottenimento della risposta utilizzando la funzione get_chat_response
res = get_chat_response(query)

# Estrazione del testo di risposta
response_text = res['result']
print("Risposta:", response_text)
```


original documents
HTML
PDF
PDF
HTML
document loader
textual documents
text splitter
chunks
Chunk of text
Chunk of text
Chunk of text
embeddings model
document embeddings
Embeddings
Embeddings
Embeddings
user question
question embedding
Embeddings
cosine similarity
vectorstore
large language model
A
answer
1
2
3
ranked results