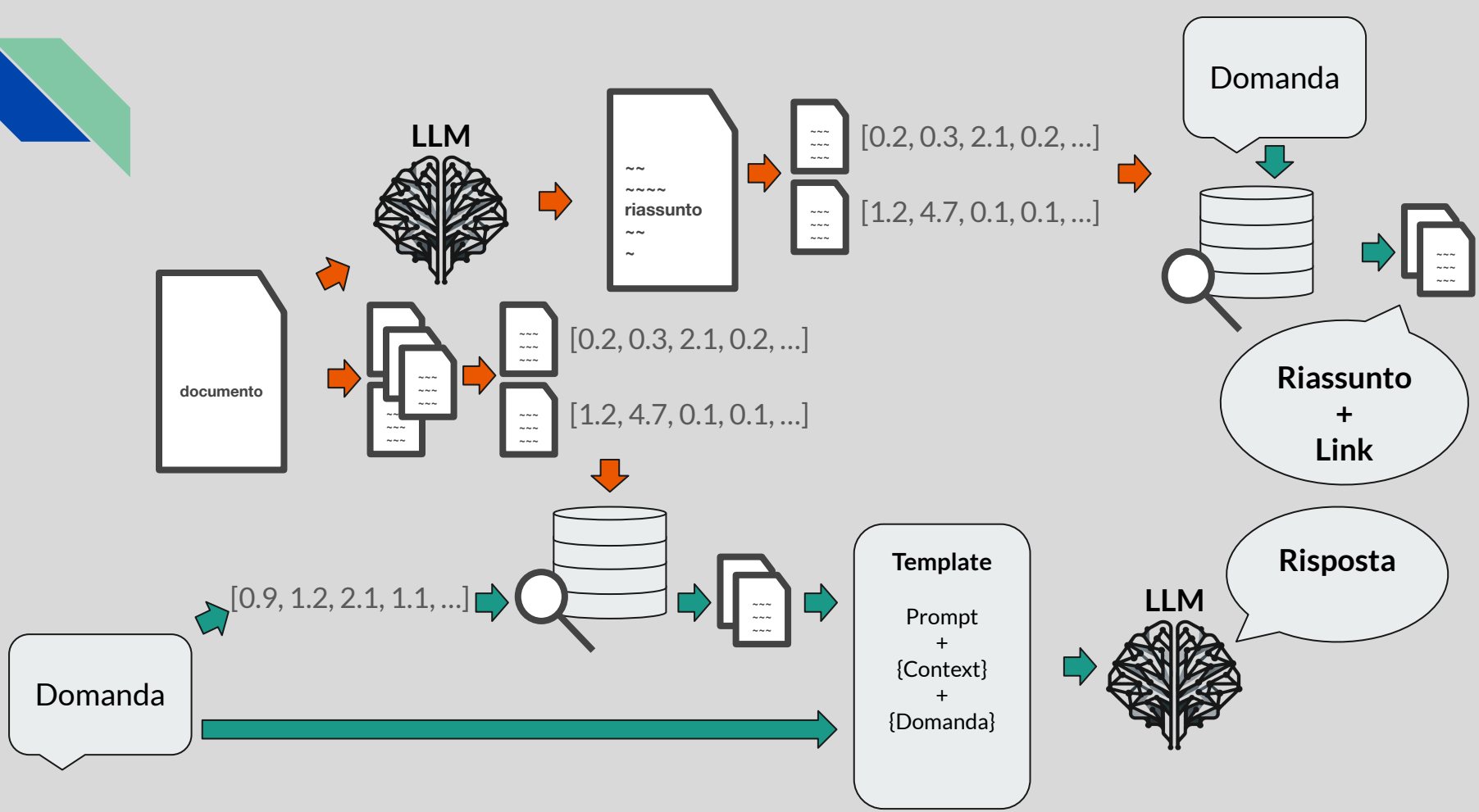
BACKEND


LLM
riassunto
[0.2, 0.3, 2.1, 0.2, ...]
[1.2, 4.7, 0.1, 0.1, ...]
Domanda
documento
[0.2, 0.3, 2.1, 0.2, ...]
[1.2, 4.7, 0.1, 0.1, ...]
Riassunto
+
Link
[0.9, 1.2, 2.1, 1.1, ...]
Domanda
Template
Prompt
+
{Context}
+
{Domanda}
LLM
Risposta


DOCUMENTO

RACCOLTA PAGINE WEB

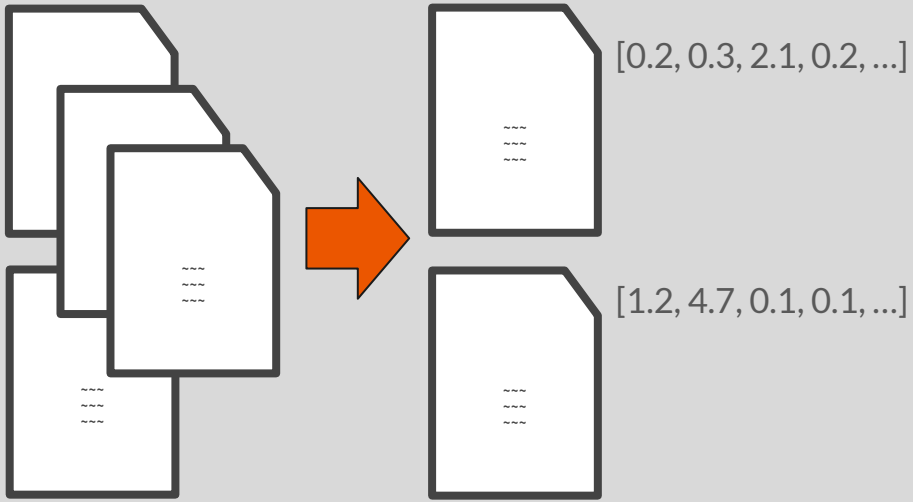
Abbiamo utilizzato un loader per importare dati dal blog creato dal gruppo 1

Funzione di un loader:

- **Accede** al link di una pagina web.
- **Esamina** il contenuto della pagina (come il testo, le immagini, i dati strutturati) e li **estrae**.

```
from langchain_community.document_loaders import
WebBaseLoader
wordpress_urls = [
        "http://karis.cloud.neth.eu/mantova-2024/",
        "http://karis.cloud.neth.eu/campania-2023/",
        "http://karis.cloud.neth.eu/firenze-05-2024/",
        "http://karis.cloud.neth.eu/grecia-2022/",
        "http://karis.cloud.neth.eu/vienna-e-monaco-2023/",
        "http://karis.cloud.neth.eu/firenze-2024/",
        "http://karis.cloud.neth.eu/bali-2023/",
        "http://karis.cloud.neth.eu/antibes-2024/",
        "http://karis.cloud.neth.eu/parma-2023/",
]
loader = WebBaseLoader(wordpress_urls)

documents = loader.load()
```


[0.2, 0.3, 2.1, 0.2, ...]
[1.2, 4.7, 0.1, 0.1, ...]

INDICIZZAZIONE

L'INDICIZZAZIONE è il processo di analisi e memorizzazione del contenuto di una pagina per renderla accessibile agli utenti quando cercano informazioni.

Funzionamento:

- I dati raccolti sono archiviati in un *indice.*
- Permette ai motori di ricerca di rispondere rapidamente alle richieste degli utenti.

Indicizzazione con gli Embeddings:

Embeddings

- Catturano il significato del testo.
- Ricerca semantica più efficace.
- Comparazione tra contenuti.

Funzionamento:

- Convertono il testo in vettori numerici.
- I vettori indicizzano e cercano documenti in modo efficiente.

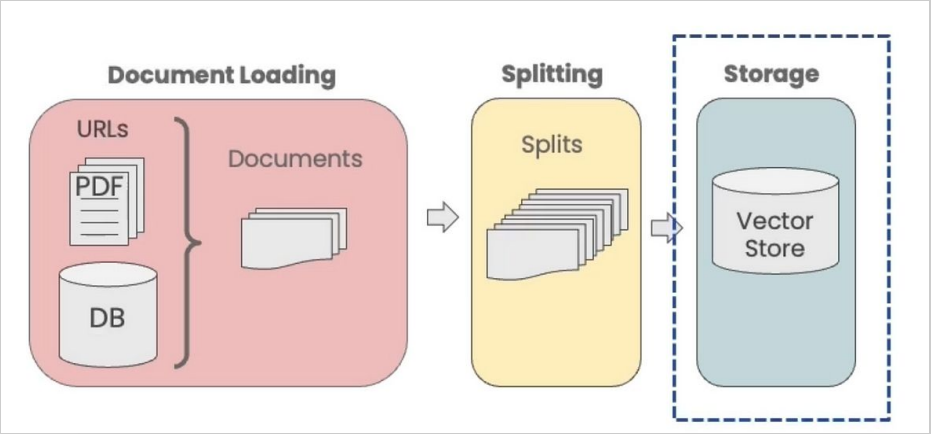

[0.2, 0.3, 2.1, 0.2, ...]
[1.2, 4.7, 0.1, 0.1, ...]

MEMORIZZAZIONE

VECTORSTORE

Che cos'è?
É un tipo di **database** persistente specializzato nella memorizzazione e ricerca di vettori numerici,che rappresentano dati complessi come testi, immagini, ecc…

Quale usiamo?
Usiamo un CHROMA perché è semplice e pratico.



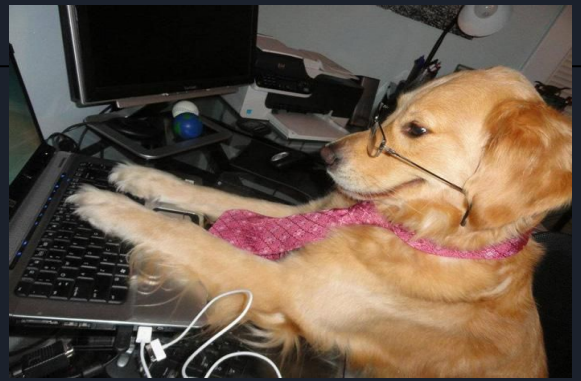

Domanda
[0.9, 1.2, 2.1, 1.1, ...]

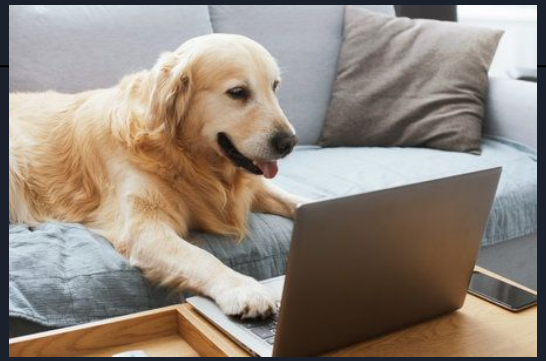
Ricerca documenti (retriever)

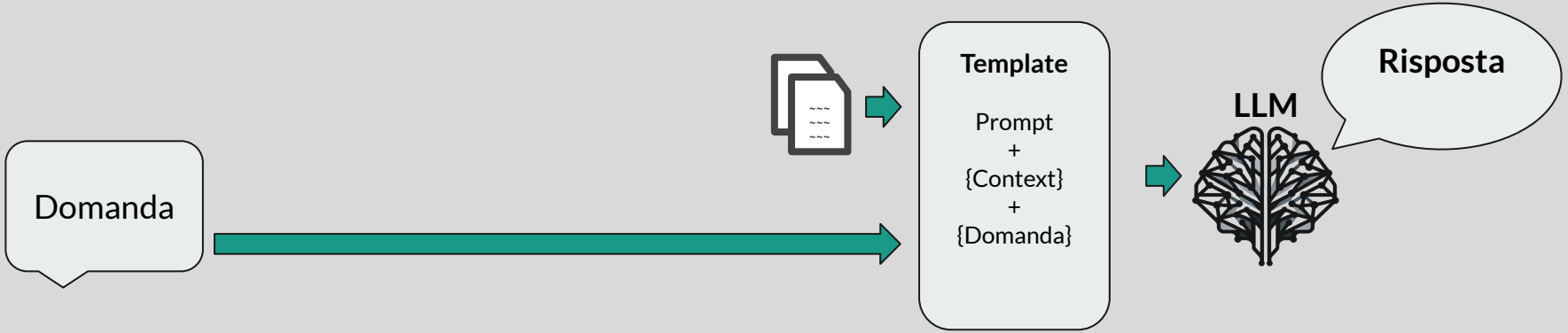
Il “retriever” è un modulo che si occupa di “estrarre” o recuperare informazioni in risposta a una richiesta.

Il *retriever* calcola la similarità tra il vettore della query e i vettori dei documenti nel Vector Store.

Successivamente, abbiamo una restituzione dei risultati grazie al parametro search_kwargs={"k": 3} -> il retriever restituisce solo i 3 documenti più simili alla query (in base alla similarità calcolata).



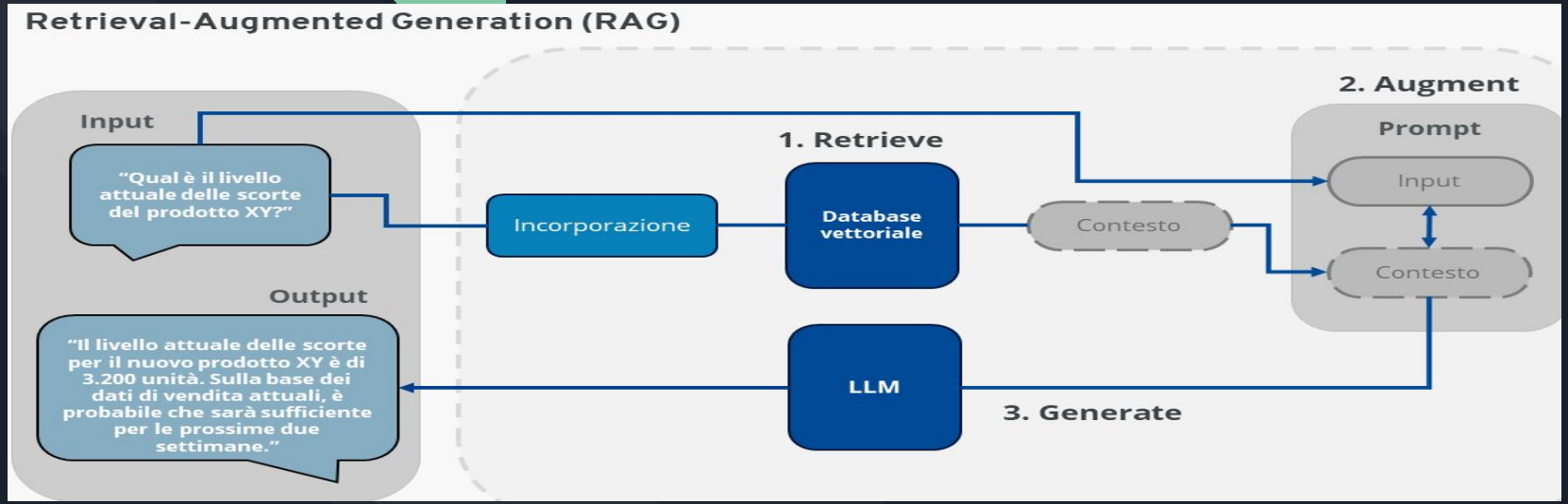



Domanda
Template
Prompt
+
{Context}
+
{Domanda}
LLM
Risposta

Cos'è la generazione delle risposte?

Il codice configura un modello GPT-4 mini (ChatOpenAI) per rispondere a una richiesta dell'utente utilizzando un template di prompt che guida la risposta.

- **Template di prompt**: Definisce regole per rispondere in modo chiaro e conciso, utilizzando solo il contesto fornito e citando le fonti.
- **Chain**: Combina il template e il modello per elaborare la domanda dell'utente con il contesto specificato.
- **Risultato**: Il sistema genera una risposta in base al testo e alla domanda, seguendo le regole stabilite (ad esempio, usare la lingua dell'utente e non inventare informazioni).



API

Le API sono il ponte tra **backend** (dove risiedono i dati e la logica dell'applicazione) e il **frontend** (l'interfaccia utente).

A che cosa servono le API?

Il backend mette a disposizione dei "punti di accesso" (endpoint API) che il frontend può utilizzare per richiedere o inviare informazioni. Questi endpoint seguono protocolli standard (come HTTP) e restituiscono risposte formattate, ad esempio, in JSON o XML.

Quando l'utente interagisce con l'interfaccia (clicca un pulsante, inserisce dati, ecc.), il frontend invia richieste al backend tramite metodi HTTP.

Metodi HTTP

1) **GET**: per recuperare dati (es. articoli di un blog)
2) **POST**: per inviare nuovi dati (es. creare un account)
3) **PUT/PATCH**: per aggiornare dati (es. modificare un profilo)
4) **DELETE**: per eliminare dati (es. cancellare un commento)

Il server elabora la richiesta, accede ai dati (magari da un database), applica la logica necessaria e restituisce una risposta al frontend.

Il frontend riceve una risposta e la utilizza per aggiornare l'interfaccia utente.

GRAZIE PER L’ATTENZIONE!