



UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

PROGETTO FACT-CHECKER

**FRANCESCO PANARIELLO M63001433
GIOVANNI RICCARDI M63001480
RICCARDO ROMANO M63001489**

Sommario

| | |
|--|----|
| 1. Analisi del Problema..... | 4 |
| 2. Strumenti utilizzati | 4 |
| 2.1 Google Colab..... | 4 |
| 2.2 MongoDB..... | 5 |
| 2.3 HuggingFace..... | 5 |
| 2.4 World News API | 6 |
| 2.5 Streamlit | 6 |
| 2.6 Local Tunnel | 7 |
| 3. Raccolta e Gestione dei Dati..... | 7 |
| 3.1 Selezione delle Fonti | 7 |
| 3.2 Raccolta degli Articoli nel DataBase | 7 |
| 3.2.1 Salvataggio ed Elaborazione degli Articoli | 8 |
| 3.2.2 Classificazione delle Emozioni | 8 |
| 3.2.3 Split dei Testi in Frasi | 8 |
| 3.2.4 Generazione degli Embeddings | 8 |
| 3.2.5 Visualizzazione e Salvataggio dei Risultati | 9 |
| 3.3 Import in MongoDB | 9 |
| 3.3.1 Connessione al Database | 9 |
| 3.3.2 Inserimento dei Dati con Logica di Retry | 9 |
| 3.3.3 Generazione degli Embeddings per le Query degli Utenti | 9 |
| 3.3.4 Ricerca Vettoriale in MongoDB | 9 |
| 4. Modelli e Tecniche Utilizzate..... | 10 |
| 4.1 Introduzione ai Modelli di Linguaggio | 10 |
| 4.2 Scelta dell'LLM | 10 |
| 4.2.1 Quantizzazione ad 8 Bit del Modello LLaMAntino | 10 |
| 4.3 Retrieval-Augmented Generation (RAG) | 11 |
| 4.3.1 Vantaggi Implementazione RAG | 11 |
| 5. Streamlit Dashboard | 12 |
| 5.1 Implementazione..... | 12 |
| 5.1.1 Precaricamento delle risorse | 12 |
| 5.1.2 Implementazione strutturale..... | 12 |
| 5.2 Interfaccia | 13 |
| 5.2.1 Question&Answer | 13 |
| 5.2.3 Fact-Checker | 13 |
| 5.2.3 ChatBot | 14 |

| | |
|--|----|
| 5.4 Analisi dei Dati e Visualizzazioni | 15 |
| 5.4.1 Word Cloud dei Titoli | 15 |
| 5.4.2 Numero di Articoli nel Dataset | 15 |
| 5.4.3 Distribuzione delle Date di Pubblicazione | 16 |
| 5.4.4 Distribuzione Articoli per Testata Giornalistica | 16 |
| 5.4.5 Distribuzione degli articoli per nome dell' autore | 17 |
| 5.4.5 Emotion in base alla Parola Chiave | 18 |

1. Analisi del Problema

Il progetto si concentra sull'implementazione di sistemi avanzati di Question-Answering (Q&A) utilizzando Large Language Models (LLM) e la tecnica Retrieval Augmented Generation (RAG). Questa combinazione consente agli LLM di accedere ad informazioni esterne e generare risposte coerenti con il contesto.

L'obiettivo è sviluppare un sistema Q&A che fornisca un insieme di report con analisi dei dati raccolti e la possibilità per gli utenti di interrogare il sistema e ricevere risposte contestualmente rilevanti.

Il progetto prevede una fase di raccolta dati grezzi da fonti esterne, seguita da una preelaborazione per prepararli all'inserimento nell'LLM e una fase di analisi dati per ottenere analytics.

I dati vengono memorizzati in un database dedicato mediante MongoDB. Il caso d'uso per il progetto in questione è il fact-checking attraverso Q&A, che sfrutta le capacità degli LLM e delle tecniche RAG per contrastare la disinformazione nel discorso politico e culturale. Verranno utilizzati articoli giornalistici come dataset per integrare le informazioni nel sistema Q&A di fact-checking, consentendo agli utenti di ricevere risposte accurate basate sulla veridicità dei fatti.

Vengono suggerite risorse aggiuntive come HuggingFace per modelli linguistici e servizi come WorldNewsAPI e NewsAPI per accedere a notizie da fonti diverse. Strumenti come Dash, Streamlit o Shiny possono essere utilizzati per creare e distribuire applicazioni di dati con interfacce personalizzate.

2. Strumenti utilizzati

2.1 Google Colab

Google Colab, è una piattaforma gratuita offerta da Google che consente di scrivere ed eseguire codice Python direttamente nel browser. È particolarmente utile per lo sviluppo e la sperimentazione di progetti di machine learning e data science. Tra le sue principali caratteristiche, Colab offre:

- **Accesso a GPU e TPU:** Permette di utilizzare unità di elaborazione grafica (GPU) e unità di elaborazione tensoriale (TPU) per accelerare i processi di calcolo, particolarmente utile per reti neurali e grandi dataset.
- **Integrazione con Google Drive:** Consente di salvare e condividere facilmente i notebook, facilitando la collaborazione tra diversi utenti.
- **Esecuzione interattiva del codice:** I notebook permettono di eseguire il codice in celle separate, rendendo possibile un approccio incrementale allo sviluppo e alla verifica del codice.

2.2 MongoDB

MongoDB è un database non relazionale orientato ai documenti, progettato per la gestione di grandi volumi di dati in modo flessibile e scalabile. Alcune caratteristiche principali di MongoDB includono:

- **Struttura a documenti:** A differenza dei database relazionali che utilizzano tabelle e righe, MongoDB memorizza i dati in documenti JSON flessibili e autocontenuti. Questo modello di dati rende più semplice e intuitivo il mapping dei dati delle applicazioni.
- **Scalabilità orizzontale:** MongoDB è progettato per scalare orizzontalmente in modo nativo, consentendo la distribuzione dei dati su più server. Ciò permette di gestire grandi volumi di dati e di gestire carichi di lavoro ad alta intensità.
- **Query potenti:** MongoDB supporta query flessibili e potenti attraverso un linguaggio di query ricco di funzionalità. È in grado di gestire query complesse e offre una vasta gamma di operatori per manipolare e filtrare i dati.
- **Aggregazioni:** MongoDB offre un framework di aggregazione potente che consente di eseguire analisi avanzate e operazioni di aggregazione sui dati memorizzati.
- **Indicizzazione efficiente:** MongoDB supporta indicizzazioni efficienti che consentono di migliorare le prestazioni delle query e delle operazioni di ricerca.
- **Replicazione e ridondanza dei dati:** MongoDB offre funzionalità native di replica e ridondanza dei dati per garantire l'affidabilità e la disponibilità dei dati.

2.3 HuggingFace

HuggingFace è una piattaforma open source che offre una vasta gamma di risorse e strumenti per il Natural Language Processing (NLP) e il machine learning basato su linguaggio. Alcune caratteristiche principali di Hugging Face includono:

- **Libreria di trasformatori (Transformers):** HuggingFace è famoso soprattutto per la sua libreria Transformers, che fornisce un'implementazione modulare e preaddestrata di una varietà di modelli basati su trasformatori per il NLP, inclusi modelli come BERT, GPT e altri. Questi modelli possono essere facilmente integrati nelle applicazioni per compiti come classificazione del testo, generazione di testo, traduzione automatica e molto altro.
- **HuggingFace Hub:** La piattaforma offre un hub centrale dove gli sviluppatori possono condividere e scoprire modelli preaddestrati, dataset e altri asset correlati al NLP. Questo favorisce la collaborazione e la condivisione di risorse all'interno della comunità.
- **Interfaccia semplice:** HuggingFace offre un'interfaccia utente semplice e intuitiva che consente agli sviluppatori di utilizzare rapidamente modelli preaddestrati per compiti specifici senza la necessità di costruire modelli da zero.

2.4 World News API

World News API è un servizio che fornisce accesso a una vasta gamma di notizie provenienti da fonti di informazione globali. Alcune caratteristiche principali di World News API includono:

- **Accesso a notizie globali:** World News API offre un'ampia copertura delle notizie provenienti da tutto il mondo, permettendo agli sviluppatori di accedere a informazioni aggiornate su una vasta gamma di argomenti e regioni geografiche.
- **Ampliamente documentato:** Il servizio è ben documentato, fornendo agli sviluppatori tutte le informazioni necessarie per integrare le notizie nei loro progetti in modo efficiente e affidabile.
- **Facilità d'uso:** World News API offre un'interfaccia di programmazione semplice e intuitiva che consente agli sviluppatori di accedere facilmente alle notizie utilizzando richieste HTTP standard.
- **Personalizzazione:** Il servizio offre opzioni per personalizzare le richieste in base alle preferenze dell'utente, consentendo di filtrare le notizie per lingua, categoria, fonte e altro ancora.
- **Affidabilità:** World News API garantisce un alto livello di affidabilità e disponibilità delle notizie, consentendo agli sviluppatori di integrare facilmente le notizie nei loro progetti senza preoccuparsi di problemi di scalabilità o disponibilità.

2.5 Streamlit

Streamlit è un framework open source che consente agli sviluppatori di creare applicazioni web interattive per l'analisi dati e la visualizzazione dei dati in modo rapido e semplice utilizzando Python. Alcune caratteristiche principali di Streamlit includono:

- **Semplicità d'uso:** Streamlit permette agli sviluppatori di creare rapidamente applicazioni web utilizzando solo codice Python. Non è richiesta alcuna conoscenza approfondita di HTML, CSS o JavaScript.
- **Ricchezza di componenti:** Streamlit offre una vasta gamma di componenti predefiniti che consentono di creare interfacce utente interattive per l'analisi e la visualizzazione dei dati, come grafici, tabelle, widget interattivi e altro ancora.
- **Aggiornamenti in tempo reale:** Le applicazioni create con Streamlit si aggiornano automaticamente ogni volta che il codice Python sottostante viene modificato, consentendo agli utenti di visualizzare immediatamente le modifiche apportate.
- **Integrazione semplice:** Streamlit può essere facilmente integrato con librerie Python popolari per l'analisi dati e il machine learning come Pandas, Matplotlib, Plotly e altri, consentendo agli sviluppatori di creare applicazioni web interattive utilizzando le loro librerie preferite.
- **Scalabilità:** Streamlit è progettato per gestire applicazioni web di grandi dimensioni e ad alto traffico, consentendo agli sviluppatori di creare e distribuire applicazioni web per un pubblico ampio e diversificato.

2.6 Local Tunnel

Localtunnel è uno strumento semplice e potente che permette di esporre un server locale su Internet tramite un URL pubblico. Alcune caratteristiche principali di Localtunnel includono:

- **Facilità d'uso:** Localtunnel è molto semplice da configurare e utilizzare, richiedendo solo pochi comandi per creare un tunnel pubblico verso il proprio server locale.
- **Supporto per più protocolli:** Localtunnel supporta sia HTTP che HTTPS, consentendo di esporre applicazioni web locali in modo sicuro.
- **URL unici e temporanei:** Ogni volta che si avvia Localtunnel, viene generato un URL unico che può essere condiviso con altri per consentire l'accesso temporaneo alla propria applicazione.
- **Compatibilità:** È compatibile con vari ambienti di sviluppo e strumenti, come Streamlit su Google Colab, permettendo di mostrare prototipi e applicazioni interattive senza configurazioni complesse.
- **Accesso remoto:** Consente di testare e mostrare applicazioni in esecuzione su un ambiente locale a collaboratori, clienti o utenti finali da qualsiasi parte del mondo.

3. Raccolta e Gestione dei Dati

Implementare un modello di Language Learning Model (LLM) con Retrieval-Augmented Generation (RAG) richiede un dataset di alta qualità che possa essere utilizzato per arricchire la conoscenza del modello. Per raggiungere questo obiettivo, sono state selezionate una serie di giornali online con un alto tasso di affidabilità e, tramite chiamate API al servizio di World News API, vengono estratti gli articoli da quest'ultime.

3.1 Selezione delle Fonti

È stato scelto un sottoinsieme di fonti giornalistiche online italiane, note per la loro affidabilità e copertura delle notizie. Ecco l'elenco delle fonti selezionate:

- **ANSA:** <https://www.ansa.it/>
- **Il Sole 24 Ore:** <https://www.ilsole24ore.com>
- **La Repubblica:** <https://www.repubblica.it>
- **Il Mattino:** <https://www.ilmattino.it>
- **La Stampa:** <https://www.lastampa.it>
- **Libero Quotidiano:** <https://www.liberoquotidiano.it>

3.2 Raccolta degli Articoli nel DataBase

Utilizzando le API di WorldNewsAPI, è possibile raccogliere articoli pubblicati in un determinato intervallo di tempo. A causa delle limitazioni economiche e temporali, la raccolta viene effettuata giornalmente con un numero limitato di chiavi API, richiedendo quindi più chiamate nel tempo. Durante questo processo, vengono specificati i parametri necessari, come le fonti di notizie, le date di pubblicazione, la lingua e il numero massimo di articoli da raccogliere.

3.2.1 Salvataggio ed Elaborazione degli Articoli

Gli articoli raccolti vengono salvati in un file JSON e successivamente caricati in un DataFrame di pandas per ulteriori elaborazioni. Il primo passo consiste nel rimuovere le colonne non necessarie, come l'ID e l'immagine, e nel filtrare gli articoli privi di contenuto. Questo è fondamentale per garantire la qualità del dataset e facilitare l'analisi e l'utilizzo dei dati.

3.2.2 Classificazione delle Emozioni

Per arricchire ulteriormente il dataset in modo da permettere un'analisi più approfondita, è stato utilizzato un classificatore di emozioni. Questo passaggio permette di aggiungere una dimensione emotiva ai dati testuali, migliorando la capacità del modello di comprendere e rispondere alle sfumature emotive presenti nei testi.

La scelta è ricaduta sull'*EmotionClassifier* della libreria *feel_it*, un modello preaddestrato specificamente per la lingua italiana.

Questo classificatore è in grado di identificare quattro emozioni fondamentali: gioia, rabbia, tristezza e paura. Innanzitutto, è stata creata un'istanza del classificatore di emozioni ed estratta la colonna 'text' dal DataFrame contenente gli articoli combinati.

Successivamente, è stato utilizzato il classificatore per prevedere l'emozione associata a ciascun testo e aggiunti i risultati come una nuova colonna.

Questo permette di avere una visione più completa delle emozioni espresse negli articoli raccolti.

3.2.3 Split dei Testi in Frasi

Per migliorare la performance della ricerca vettoriale all'interno del database, sono stati suddivisi i testi degli articoli in singole frasi.

Il motivo principale di questa suddivisione è legato alla lunghezza dei testi. Quando i testi sono troppo lunghi, possono risultare difficili da gestire nel prompt, causando inefficienze e problemi di performance nel modello. Splittando i testi in frasi, si ottiene un'unità di testo più gestibile e precisa, che possono essere analizzate e utilizzate più facilmente.

Utilizzando la libreria NLTK, viene creata una funzione per splittare il testo in frasi e generati nuovi record per ciascuna frase, mantenendo tutte le altre informazioni rilevanti (come titolo, summary, URL, data di pubblicazione, autore, lingua, paese di origine ed emozione).

3.2.4 Generazione degli Embeddings

È stato utilizzato un modello preaddestrato di Sentence Transformers (nickprock/sentence-bert-base-italian-xxl-uncased) per creare gli embeddings dei testi. Questo modello è particolarmente adatto per testi in lingua italiana.

Gli embeddings sono stati generati per ciascuna frase estratta e aggiunti al DataFrame come una nuova colonna. Questo passaggio è cruciale per preparare i dati, consentendo una più rapida ricerca semantica all'interno del DataBase. Si noti inoltre che il modello utilizza una dimensione vettoriale dei dati pari a 768.

3.2.5 Visualizzazione e Salvataggio dei Risultati

Infine, prima del caricamento nel DataBase è stato salvato il DataFrame aggiornato in un file CSV per facilitarne l'uso futuro. Questo file contiene tutte le informazioni necessarie per l'implementazione e l'espansione della conoscenza del modello LLM con RAG, offrendo un dataset ricco, accurato e aggiornato.

3.3 Import in MongoDB

Per l'elaborazione del progetto è stato scelto di impiegare il database MongoDB per la gestione dei dati utilizzati. La scelta di quest'ultimo è ricaduta in merito alla possibilità di usufruire un'ottima flessibilità nella gestione dei dati consentendo dunque di memorizzare dati strutturati, semi-strutturati e non strutturati senza la necessità di schemi rigidi. Inoltre, l'utilizzo di un Cluster MongoDB garantisce un'elevata scalabilità orizzontale, il che significa la possibilità di gestire grandi volumi di dati senza rinunciare alle prestazioni.

Un ulteriore aspetto significativo è quello della ricerca vettoriale mediante Atlas Search. Questa consente di eseguire query di ricerca su campi indicizzati in modo veloce, preciso ed intuitivo.

3.3.1 Connessione al Database

Per memorizzare i dati elaborati, avviene la connessione al database MongoDB. Viene utilizzato un client con *timeouts* estesi per garantire che la connessione sia stabile anche in presenza di latenza. Dopo aver stabilito la connessione, viene verificato il successo mediante l'invio di un ping al server.

3.3.2 Inserimento dei Dati con Logica di Retry

L'inserimento dei dati nel database viene gestito in batch, utilizzando una logica di retry per garantire che ogni batch venga inserito correttamente. Questo approccio è fondamentale per gestire eventuali interruzioni di connessione o altri errori temporanei. È stato impostato un numero massimo di tentativi di inserimento per ciascun batch e, in caso di errore, vengono attesi alcuni secondi prima di riprovare.

3.3.3 Generazione degli Embeddings per le Query degli Utenti

Per permettere la ricerca vettoriale, vengono generati gli embeddings per le query degli utenti utilizzando lo stesso modello di embedding usato per i testi degli articoli. Questo garantisce che le query e i documenti siano rappresentati nello stesso spazio vettoriale, permettendo il confronto e la ricerca.

3.3.4 Ricerca Vettoriale in MongoDB

La ricerca vettoriale è eseguita tramite una pipeline di MongoDB che:

- Ricerca i documenti con embeddings simili a quello della query dell'utente tramite la similarità coseno.

- Ordina i risultati per data di pubblicazione in ordine decrescente per mostrare gli articoli più recenti in cima selezionandone i più rilevanti.
- Proietta solo i campi rilevanti (titolo, testo, summary, URL, data di pubblicazione) ed il punteggio della ricerca.

4. Modelli e Tecniche Utilizzate

4.1 Introduzione ai Modelli di Linguaggio

I modelli di linguaggio di grandi dimensioni (LLMs) sono strumenti potenti che possono essere utilizzati per una varietà di compiti di elaborazione del linguaggio naturale, tra cui la generazione di testo, la traduzione, il completamento del testo, e molto altro.

Tuttavia, uno dei principali svantaggi di questi modelli è la loro dimensione, che può rendere difficile il loro utilizzo su hardware con risorse limitate.

4.2 Scelta dell'LLM

Sono stati testati diversi approcci per caricare e utilizzare un modello di linguaggio italiano adatto alle esigenze del problema. Ecco una sintesi dei tentativi e dei risultati ottenuti:

- **LLaMAntino-3-ANITA-8B-Inst-DPO-ITA:** Questo modello, sebbene potente e ben addestrato per l'italiano, è risultato troppo grande per essere caricato direttamente su hardware standard senza ottimizzazioni.
- **gpt2-small-italian:** Un modello più piccolo che poteva essere caricato, ma con performance significativamente inferiori, rendendo l'esperienza complessiva insoddisfacente.
- **gemma-2b-it:** Un modello di dimensioni intermedie, ma con una copertura limitata dell'italiano, che non soddisfaceva pienamente le esigenze linguistiche.

4.2.1 Quantizzazione ad 8 Bit del Modello LLaMAntino

Per risolvere i problemi di dimensione e risorse, si è optato per una tecnica di quantizzazione ad 8 bit. La quantizzazione è un processo che riduce la precisione dei numeri nei calcoli del modello, riducendo così la memoria richiesta e aumentando la velocità di inferenza, con una perdita minima di precisione e qualità del modello.

La quantizzazione del modello LLaMAntino a 8 bit dunque, ha portato diversi vantaggi:

- **Riduzione della Memoria:** Il modello quantizzato occupa significativamente meno spazio in memoria, rendendo possibile il suo utilizzo su hardware meno potente.
- **Miglioramento della Velocità di Inference:** Riducendo la precisione, le operazioni matematiche possono essere eseguite più velocemente, migliorando il tempo di risposta del modello.
- **Mantiene una Buona Precisione:** Nonostante la riduzione della precisione numerica, il modello mantiene una buona comprensione e generazione del linguaggio italiano, soddisfacendo i requisiti del problema.

4.3 Retrieval-Augmented Generation (RAG)

Il Retrieval-Augmented Generation (RAG) è una tecnica avanzata che combina la generazione di testo basata su modelli linguistici (Language Models, LLM) con il recupero di informazioni da fonti esterne. In sostanza, RAG unisce le capacità di generazione del linguaggio naturale con la capacità di recuperare documenti pertinenti da un grande corpo di dati, come un database di articoli, documenti di ricerca o altre fonti di informazioni. La tecnica opera mediante due step:

1. **Recupero (Retrieval):** Quando viene posta una domanda, un componente di recupero di informazioni cerca nei documenti esterni per trovare quelli più rilevanti.
2. **Generazione (Generation):** I documenti recuperati vengono poi utilizzati come contesto per un modello di generazione del linguaggio, che genera una risposta basata sia sulla domanda che sui documenti recuperati.

4.3.1 Vantaggi Implementazione RAG

- **Accesso a Informazioni Aggiornate:**
 - Uno dei principali vantaggi del RAG è la capacità di accedere a informazioni aggiornate che non sono state incluse nel set di addestramento del modello. Questo è particolarmente utile per il fact-checking e altre applicazioni che richiedono conoscenze aggiornate.
- **Risposte Più Accurate:**
 - Combinando le capacità di generazione con fonti esterne di informazione, RAG può produrre risposte più precise e informative. Il modello non si basa solo su ciò che ha imparato durante l'addestramento, ma può attingere da una vasta gamma di documenti pertinenti.
- **Flessibilità e Scalabilità:**
 - RAG può essere facilmente aggiornato e scalato. Aggiungere nuovi documenti al corpus di recupero è semplice e non richiede un nuovo addestramento del modello generativo. Questo rende il sistema altamente flessibile e adattabile a nuove informazioni e cambiamenti.
- **Riduzione del Bias:**
 - Utilizzando un "corpo" esterno e dinamico di documenti, RAG può potenzialmente ridurre i bias che possono essere presenti nei modelli linguistici pre-addestrati. Il recupero di informazioni da diverse fonti può aiutare a fornire risposte più equilibrate e rappresentative.

5. Streamlit Dashboard

In questo capitolo, viene esaminata la dashboard sviluppata grazie all' utilizzo di Streamlit, una libreria Python per la creazione di applicazioni web interattive e data-driven. La dashboard è progettata per eseguire operazioni di **Question & Answer**, **Fact-Checking** e funzionalità di **ChatBot**.

Per effettuare un bridge tra Google Colab e Streamlit è stato necessario l'utilizzo di Localtunnel, che permette di esporre l'applicazione Streamlit in esecuzione su Colab attraverso un URL pubblico accessibile da qualsiasi browser.

Ogni componente sfrutta strumenti avanzati come HuggingFace per i modelli di linguaggio, MongoDB per la gestione dei dati e delle query, embeddings multilingue per l'elaborazione del testo e la tecnica RAG (Retrieval-Augmented Generation) per ottenere le ultime notizie.

Nelle diverse sezioni all'LLM è attribuito un sys differente, ovvero un insieme di istruzioni di sistema che impostano il contesto operativo del modello in base alla propria funzionalità.

5.1 Implementazione

5.1.1 Precaricamento delle risorse

Per ottimizzare le prestazioni della dashboard e migliorare l'esperienza utente, è stata implementata una strategia di precaricamento delle risorse utilizzando la funzionalità di cache di Streamlit. Il decoratore “`@st.cache_resource`” è stato utilizzato per memorizzare in cache diverse risorse essenziali, riducendo così i tempi di caricamento e migliorando l'efficienza complessiva del sistema. Tra le risorse precaricate sono incluse la connessione al database, che viene stabilita una sola volta e riutilizzata per tutte le successive richieste. Inoltre, è stato precaricato il modello di linguaggio di grandi dimensioni (LLM) utilizzato per le elaborazioni linguistiche, permettendo un accesso rapido e immediato senza dover ricaricare il modello ad ogni aggiornamento della sessione. Infine, alcune query frequenti e critiche sono state memorizzate in cache, garantendo che i risultati siano prontamente disponibili senza necessità di esecuzione ripetuta.

5.1.2 Implementazione strutturale

Le tre funzionalità di Q&A, Fact Checking e Chat, sono studiate e strutturate in questo modo:

- **Input Utente:** Gli utenti possono inserire l' input in un campo di testo interattivo fornito dalla dashboard. Streamlit facilita la raccolta di input in modo semplice e user-friendly.
- **Elaborazione:** Il testo della domanda viene processato e passato al modello. Questo modello, basato su tecnologie di linguaggio naturale, è in grado di comprendere il contesto e la semantica delle domande poste.
 - **Sys:** L'LLM deve rispondere in maniera chiara ed esaustiva all' input dell' utente, per questo sono stati implementati diversi messaggi di sistema personalizzati per la funzionalità da offrire.
 - **Embeddings:** Vengono utilizzati embeddings per rappresentare gli input in uno spazio vettoriale, migliorando la comprensione semantica e la ricerca nel database, delle notizie correlate.
 - **Database MongoDB:** Gli embeddings e gli articoli sono memorizzati e gestiti tramite MongoDB, per permettere un accesso rapido e una gestione efficiente dei dati.

- **Tecnica RAG:** La tecnica Retrieval-Augmented Generation (RAG) viene utilizzata per arricchire gli input del modello con informazioni aggiornate, recuperando le ultime notizie rilevanti dal database.
- **Output:** La risposta generata dal modello viene mostrata sulla dashboard, fornendo un feedback immediato all'utente. Streamlit consente di aggiornare dinamicamente il contenuto della pagina.

5.2 Interfaccia

5.2.1 Question&Answer

La sezione **Question & Answer** della dashboard permette agli utenti di porre domande e ottenere una risposta argomentata dall'assistente, con riferimento agli articoli raccolti.

È stata intesa come l'interfaccia più semplice da utilizzare poiché è necessario solamente inserire l'interrogativo da valutare.

Question & Answer

Inserisci Domanda:

C'è un conflitto in Ucraina?

Sì, c'è un conflitto in Ucraina, iniziato il 24 febbraio 2022, quando la Russia ha iniziato a invadere l'Ucraina, e che attualmente prosegue.

5.2.3 Fact-Checker

La sezione **Fact-Checker** è progettata per verificare la veridicità delle notizie o articoli inseriti dagli utenti.

L'idea è quella di ottenere un responso di veridicità semplice e diretto, in base ai risultati della ricerca.

Fact Checker

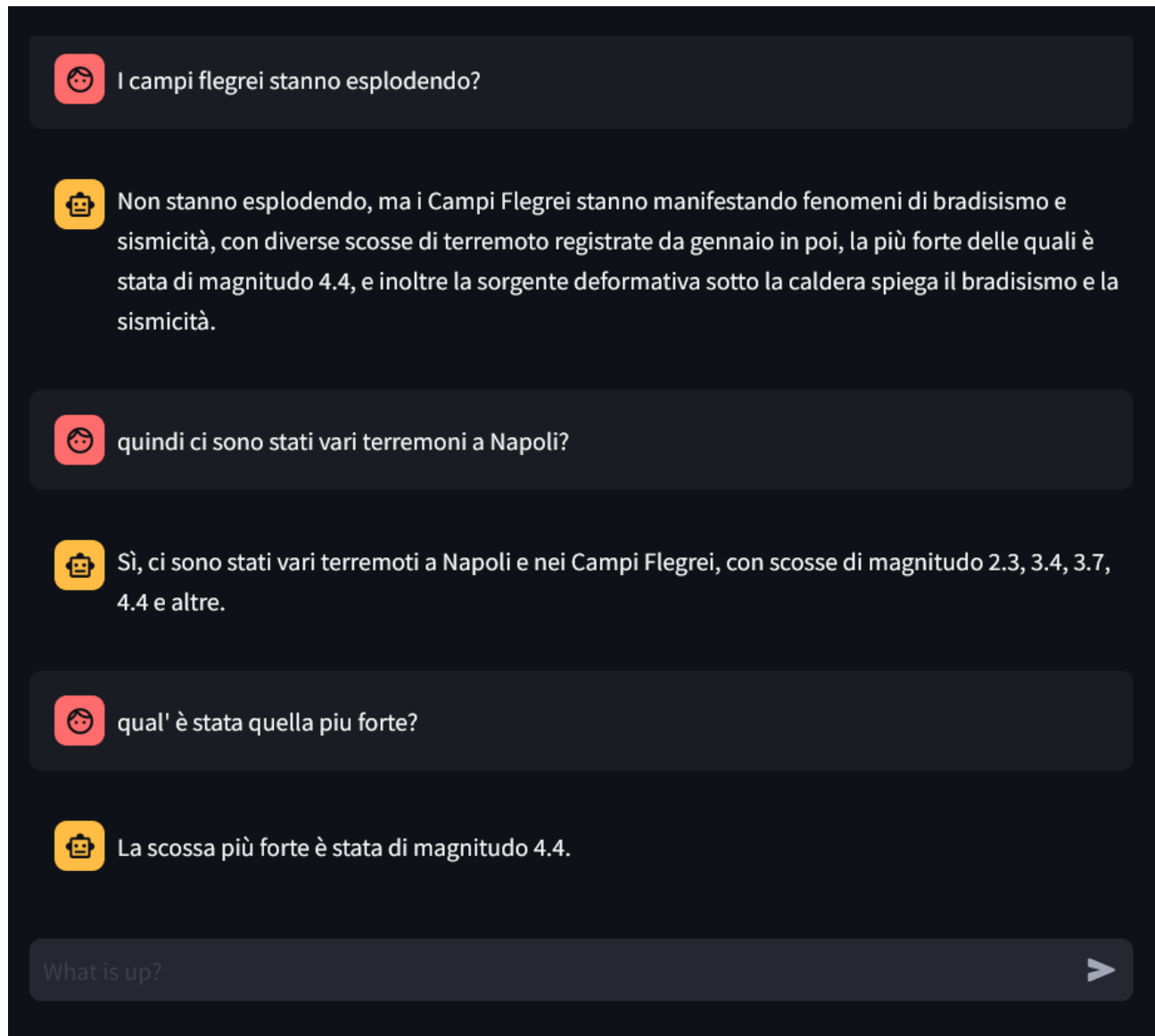
Inserisci Notizia:

Rissa in parlamento tra i parlamentari a giugno

La notizia è vera. Il 14 giugno 2022, in effetti, si è verificata una rissa in aula alla Camera dei deputati italiana, in cui il deputato del Movimento 5 Stelle Leonardo Donno è stato colpito e portato via in sedia a rotelle, e il deputato della Lega Igor Lezzi è stato accusato di averlo colpito con dei pugni ripetutamente alla testa.

5.2.3 ChatBot

La sezione **ChatBot** offre un'interfaccia interattiva in cui gli utenti possono avere una conversazione continua con un assistente virtuale, capace di memorizzare il contesto della conversazione e rispondere in modo più coerente alle nuove domande dell'utente.



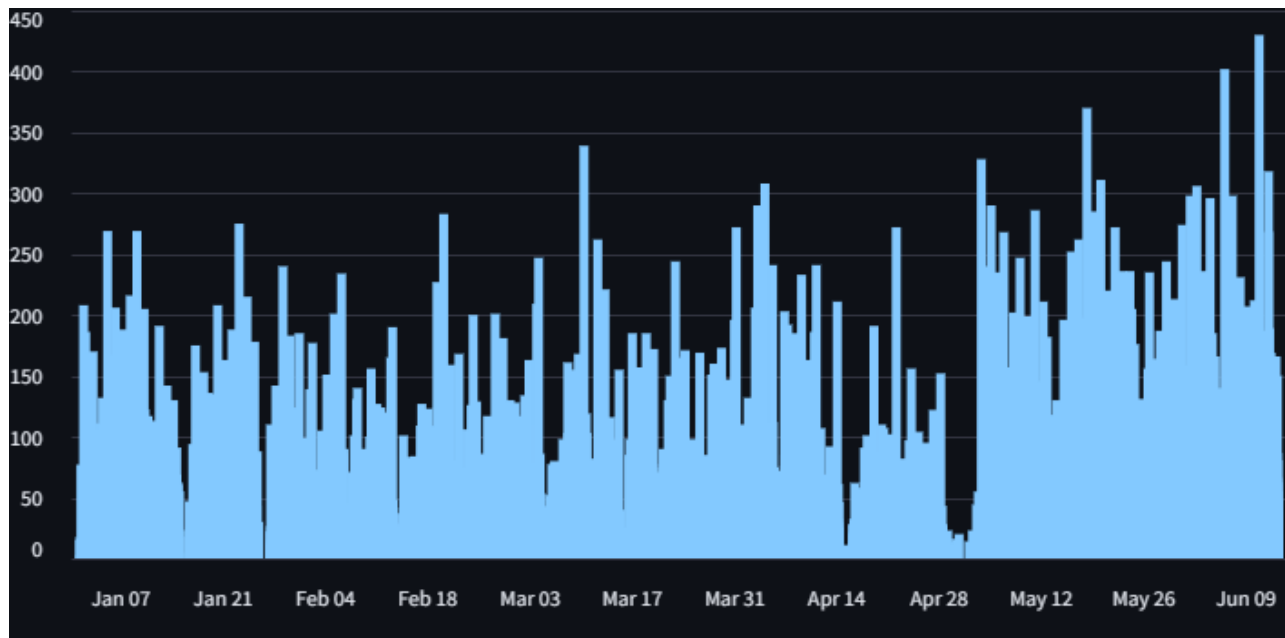
The screenshot displays a chatbot interface with a dark background. It shows a sequence of user questions and bot responses. Each message is preceded by a small icon: a red circle with a white face for user messages and a yellow circle with a white robot head for bot responses. The conversation is as follows:

- User:** I campi flegrei stanno esplodendo?
- Bot:** Non stanno esplodendo, ma i Campi Flegrei stanno manifestando fenomeni di bradisismo e sismicità, con diverse scosse di terremoto registrate da gennaio in poi, la più forte delle quali è stata di magnitudo 4.4, e inoltre la sorgente deformativa sotto la caldera spiega il bradisismo e la sismicità.
- User:** quindi ci sono stati vari terremoti a Napoli?
- Bot:** Sì, ci sono stati vari terremoti a Napoli e nei Campi Flegrei, con scosse di magnitudo 2.3, 3.4, 3.7, 4.4 e altre.
- User:** qual' è stata quella piu forte?
- Bot:** La scossa più forte è stata di magnitudo 4.4.

At the bottom of the interface, there is a text input field containing the placeholder text "What is up?" and a grey arrow button pointing to the right.

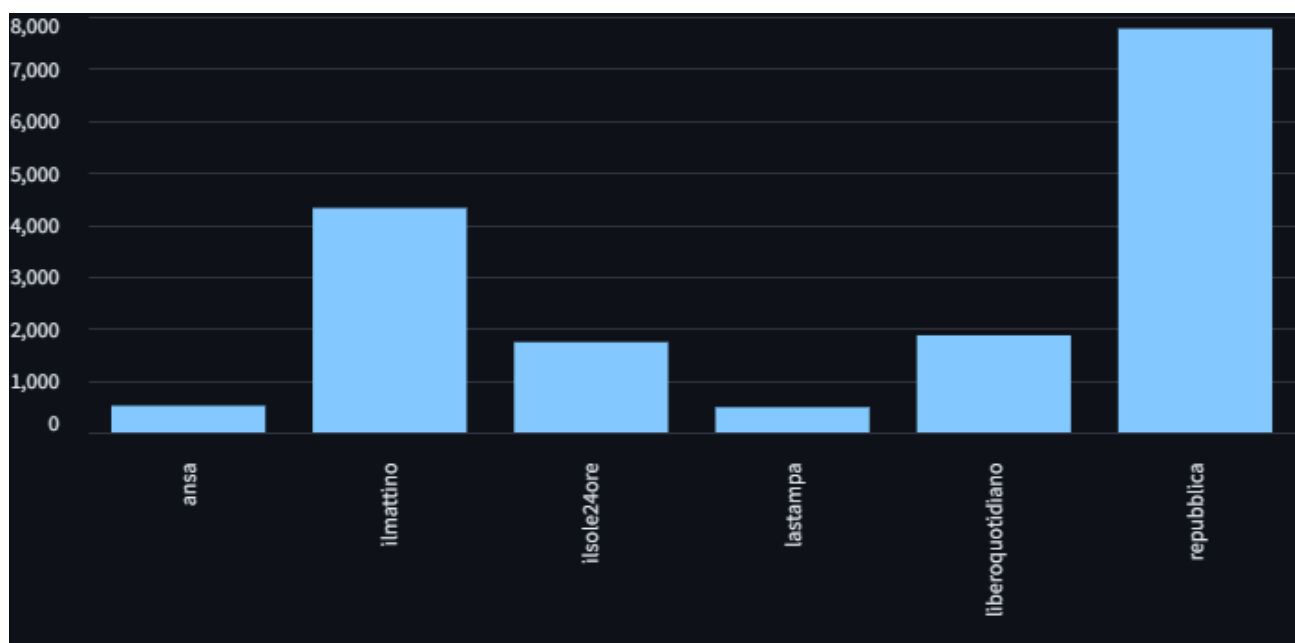
5.4.3 Distribuzione delle Date di Pubblicazione

È stato scelto di visualizzare la Distribuzione delle Date di Pubblicazione attraverso un grafico a barre per identificare e analizzare le tendenze temporali nel numero di articoli pubblicati nel corso del tempo. Questo tipo di visualizzazione è stato utile per evidenziare picchi di attività giornalistica e per identificare periodi di maggiore o minore produzione di contenuti informativi.



5.4.4 Distribuzione Articoli per Testata Giornalistica

Al fine di analizzare la distribuzione degli articoli tra le diverse testate giornalistiche, è stato implementato un grafico a barre che mostra il Numero di Articoli per Testata Giornalistica. Questa scelta è stata motivata dalla necessità di valutare la rappresentatività delle fonti di informazione nello studio, offrendo così una panoramica chiara e completa delle fonti di dati analizzate.



5.4.5 Distribuzione degli articoli per nome dell' autore

Per valutare la quantità di articoli scritti dagli autori presenti nel dataset è stato necessario applicare un insieme di algoritmi per estrapolare i nomi di persona dalla sezione Autore delle API giornalistiche. Questo perché quella colonna spesso è erroneamente riempita con informazioni non inerenti o con un prologo di informazione di riferimento al giornale. In particolare, sono stati usati due librerie per la rilevazione dei nomi di persona italiani Spacy e regex, garantendo così un'accurata estrazione dei dati e una rappresentazione efficace del lavoro giornalistico all'interno delle testate analizzate.



5.4.5 Emotion in base alla Parola Chiave

Per valutare la distribuzione delle emotions che potrebbero suscitare gli articoli in base ad una parola chiave, è stato implementato il grafico sottostante.

Tramite una query al database vengono estratti i testi che contengono la parola chiave specificata e conteggiato quante volte diverse emozioni sono espresse in relazione a quella parola chiave.

