UNIVERSITÀ DEGLI STUDI DI MILANO-BICOCCA
DIPARTIMENTO DI INFORMATICA, SISTEMISTICA E COMUNICAZIONE
CORSO DI LAUREA IN INFORMATICA

Costruzione di un database sugli eventi attraverso l'utilizzo di un webcrawler e LLMs

Relatore: Andrea Maurino
Co-relatore: Blerina Spahiu

Relazione della prova finale di:
AGOSTINO VIGANI
MATRICOLA 886517

Anno Accademico 2023 - 2024

Abstract

La tesi ha come obiettivo l'implementazione di un sistema automatico per la raccolta e l'analisi dei dati relativi agli eventi nella città di Milano, attraverso l'utilizzo di tecniche avanzate di web crawling e tecnologie innovative come i Large Language Models (LLM). Il progetto mira a risolvere il problema della frammentazione delle informazioni online utilizzando un sistema in grado di estrarre e salvare informazioni strutturate da diverse fonti web, in modo da migliorare la gestione e l'accessibilità dei dati. Il lavoro si concentra sulla realizzazione di un'infrastruttura facilmente aggiornabile che possa offrire un punto di partenza per ulteriori sviluppi e applicazioni future.

Indice

Elenco delle figure

Elenco delle tabelle

Elenco degli acronimi

LLM Large Language Model

GPT Generative Pretrained Transformer

URL Uniform Resource Locator

HTTP HyperText Transfer Protocol

HTML HyperText Markup Language

JSON JavaScript Object Notation

API Application Programming Interface

1 Introduzione

L'introduzione offre una panoramica dei benefici che può portare la costruzione di un database sugli eventi per la città di Milano, sviluppato utilizzando un approccio innovativo che combina web crawling e Large Language Models per l'estrazione dei dati dal web. Gli obiettivi della tesi includono l'acquisizione di familiarità con le tecniche di scraping, l'implementazione di un web crawler efficiente, l'utilizzo di LLM per l'analisi dei dati e la gestione centralizzata delle informazioni sugli eventi.

1.1 Ipotesi di partenza

Nel panorama culturale e sociale di una città, gli eventi giocano un ruolo fondamentale nel promuovere l'intrattenimento, la cultura e la partecipazione collettiva. Sia che si tratti di concerti, mostre d'arte, fiere o manifestazioni culturali, gli eventi offrono opportunità di incontro e di arricchimento per i cittadini e i visitatori. Essi non solo migliorano la vita sociale, ma contribuiscono anche all'economia locale attirando turisti e investimenti in tutti i settori.

L'era digitale ha portato ad una proliferazione di informazioni online sugli eventi, con numerosi siti web, social media e altre piattaforme che pubblicizzano e organizzano queste attività. Tuttavia, questa ricchezza di informazioni può anche rappresentare una sfida per chi cerca di tenersi aggiornati sugli eventi in corso. La frammentazione delle fonti e la varietà dei formati di presentazione delle informazioni rendono complesso raccogliere e utilizzare efficacemente questi dati.

In questo contesto, la costruzione di un database dedicato agli eventi assume un'importanza cruciale e il lavoro di tesi si inserisce in un ambito altamente rilevante, in quanto affronta la crescente necessità di gestire in modo efficace la dispersione delle informazioni sui fenomeni in una grande città come Milano. La raccolta e l'analisi di dati sugli eventi culturali e sociali rappresentano una sfida non solo tecnica ma anche strategica, con implicazioni rilevanti per la pianificazione urbana, il marketing e la partecipazione dei cittadini. Il progetto esplora tecniche di web crawling, affiancate dall'uso innovativo di Large Language Models (LLM), per migliorare la precisione e la strutturazione delle informazioni estratte. L'obiettivo formativo di questa tesi è duplice: acquisire competenze avanzate per la progettazione di database e padroneggiare l'uso di LLM per l'analisi dei dati estratti tramite tecniche di scraping. Questo progetto si distingue per il potenziale di creare un sistema che non solo centralizzi le informazioni, ma che le renda utilizzabili per sviluppare nuove applicazioni di intelligenza artificiale, apportando benefici concreti alla città e alla sua comunità.

Un database contenente informazioni accurate e aggiornate sugli eventi offre numerosi vantaggi: innanzitutto ha un valore inestimabile dal punto di vista analitico e strategico. Le informazioni raccolte possono essere utilizzate per l'analisi dei punti di interesse e dei comportamenti degli utenti. Ad esempio, si possono identificare le tendenze di partecipazione, i tipi di eventi più popolari e i periodi dell'anno con maggiore affluenza. Questi dati sono fondamentali per gli organizzatori di eventi e per le amministrazioni cittadine, che possono utilizzarli per pianificare meglio le loro attività e per allocare le risorse in modo più efficace.

Inoltre, dei dati ben strutturati sono una risorsa preziosa per la costruzione di modelli di intelligenza artificiale che richiedono grandi quantità di informazioni. Gli algoritmi di machine learning possono essere addestrati per prevedere l'affluenza di determinati luoghi e strade in ambito di smart mobility, per personalizzare le raccomandazioni agli utenti e per ottimizzare le campagne di marketing. La disponibilità di dati accurati e completi consente di sviluppare applicazioni innovative che migliorano la gestione degli eventi e l'esperienza dei partecipanti.

Un altro aspetto rilevante è la possibilità di utilizzare il database per monitorare l'impatto sociale ed economico degli eventi. Attraverso l'analisi dei dati, è possibile valutare come gli eventi influenzano la vita della città, sia in termini di benefici economici che in termini di coesione sociale. Questa conoscenza è essenziale per giustificare investimenti pubblici e privati nel settore degli eventi e per promuovere politiche culturali efficaci.

In conclusione, la costruzione di un database sugli eventi non è solo un esercizio tecnico, ma un'iniziativa con potenziali benefici significativi per la comunità. La centralizzazione e la strutturazione delle informazioni sugli eventi facilitano l'accesso, l'analisi e l'utilizzo dei dati, migliorando sia l'esperienza degli utenti che l'analisi complessiva degli avvenimenti.

1.2 Obiettivi della Tesi

L'obiettivo principale della tesi è la raccolta di dati sugli eventi per la città di Milano, tramite l'integrazione di un web crawler e tecnologie innovative come i large language models (LLM). Questo progetto mira a realizzare un sistema innovativo che possa raccogliere, analizzare e strutturare le informazioni sugli eventi in modo accurato ed efficiente. Attraverso questo lavoro si intende fornire un servizio utile che possa supportare analisi avanzate e lo sviluppo di applicazioni di intelligenza artificiale.

Il web crawling è una tecnica utilizzata per automatizzare la raccolta di dati dai siti web. Un web crawler, noto anche come spider, è un programma che naviga automaticamente tra le pagine web, estraendo informazioni secondo regole predefinite. Questo processo è fondamentale per raccogliere dati di qualsiasi tipo da fonti online, permettendo di creare archivi aggiornate e completi. Uno degli obiettivi formativi principali è acquisire una conoscenza approfondita di queste tecniche e della loro applicabilità in diversi contesti. Questo comprende l'analisi dei principi fondamentali del crawling, la capacità di progettare e implementare un web crawler efficiente e la gestione delle sfide legate all'estrazione automatica di dati da fonti web eterogenee. Attraverso l'esperienza pratica, si acquisirà una competenza approfondita che può essere applicata a molteplici problemi di raccolta dati.

Un altro obiettivo fondamentale è lo sviluppo di competenze necessarie per interagire con i Large Language Model (LLM). Essi sono modelli di intelligenza artificiale basati su tecniche avanzate di machine learning, in particolare su reti neurali di grandi dimensioni, addestrati su vaste quantità di testo. Questi modelli sono in grado di comprendere, generare e analizzare il linguaggio naturale, permettendo di svolgere compiti complessi come l'estrazione di informazioni strutturate, la traduzione automatica e la risposta a domande. Grazie alla loro capacità di interpretare il contesto, gli LLM migliorano l'accuratezza e la precisione nell'analisi dei dati testuali. Questi modelli offrono potenti strumenti per l'analisi del testo e l'estrazione di informazioni strutturate. L'obiettivo è comprendere come configurare e utilizzare un LLM per analizzare i dati, migliorando l'accuratezza e la precisione delle informazioni ottenute. Questo include la capacità di processare i dati, configurare il modello per specifiche esigenze e interpretare i risultati in modo efficace.

L'intento è progettare una struttura di database che sia non solo capace di gestire grandi volumi di dati in maniera efficace, ma anche facilmente accessibile e aggiornabile. Tra gli obiettivi concreti e misurabili del progetto vi è l'estrazione di informazioni chiave indispensabili sugli eventi, come il *titolo*, la *data* (in un formato standardizzato), il *luogo* e, opzionalmente, una *descrizione* e altre informazioni aggiuntive. Questi dati costituiscono la base essenziale per una corretta rappresentazione degli eventi, e devono essere raccolti con

precisione in modo da poter essere utilizzati agevolmente per analisi successive o applicazioni esterne.

Inoltre, uno scopo trasversale di questa tesi è l'analisi e la valutazione dei risultati ottenuti, garantendo sia la qualità dei dati estratti sia la performance complessiva del sistema. Questo implica lo sviluppo di metodi per monitorare e valutare la correttezza e la completezza delle informazioni raccolte, con l'intento di migliorare continuamente l'efficacia del processo di estrazione. I dati estratti devono seguire uno *schema ben definito*, che sarà deciso e controllato per garantire la coerenza e la standardizzazione delle informazioni. La verifica della conformità a questo schema rappresenterà un importante indicatore della qualità del sistema, assicurando che i dati siano strutturati in maniera uniforme e facilmente accessibili per future integrazioni o aggiornamenti. L'obiettivo è sviluppare una metodologia di valutazione che possa identificare punti di forza e aree di miglioramento, garantendo che il sistema finale non solo soddisfi i requisiti iniziali, ma sia anche in grado di adattarsi e migliorarsi continuamente.

In sintesi, questa tesi si pone l'obiettivo di combinare teoria e pratica in un progetto che unisce web crawling, intelligenza artificiale e gestione dei dati. Attraverso questo percorso, non solo verranno acquisite competenze tecniche innovative e avanzate, ma si contribuirà anche alla creazione di un sistema con potenziali applicazioni in molti altri contesti urbani e culturali.

1.3 Struttura della Tesi

La tesi è suddivisa nei seguenti capitoli, ciascuno dei quali tratta un aspetto fondamentale del progetto, dalla definizione del contesto fino alla valutazione dei risultati ottenuti:

- **Capitolo 1: Introduzione** – Introduce l'argomento della tesi, il problema affrontato, gli obiettivi e la metodologia utilizzata.
- **Capitolo 2: Stato dell'arte** – Fornisce una revisione della letteratura sulle tecniche e i sistemi attuali basati su web crawling e Large Language Models, utilizzati per la gestione di dati eterogenei.
- **Capitolo 3: Tecnologie abilitanti** – Analizza in dettaglio le tecnologie scelte per lo sviluppo del sistema, con un focus particolare su web crawling e LLM.
- **Capitolo 4: Implementazione del codice** – Descrive il processo di sviluppo del sistema, dall'idea all'implementazione del codice.

- **Capitolo 5: Esperimenti e risultati** – Presenta i risultati ottenuti attraverso l'implementazione del sistema, compresa una valutazione quantitativa e qualitativa delle prestazioni.
- **Capitolo 6: Conclusioni** – Riassume i risultati della tesi e suggerisce possibili miglioramenti e sviluppi futuri.

2
Stato dell'arte

In questo capitolo verrà esaminato come altri autori e progetti hanno affrontato il problema della costruzione di sistemi di web crawling che utilizzano Large Language Models (LLMs). Verranno analizzati vari approcci e tecnologie adottati, mettendo in luce i loro punti di forza e le limitazioni, per poi discutere come il presente lavoro si distingue da questi tentativi, avanzando lo stato dell'arte.

2.1 Sistemi di Web Crawling con LLM

Negli ultimi anni, l'integrazione di modelli di linguaggio di grandi dimensioni (LLMs) nei sistemi di web crawling ha attirato un crescente interesse. Questi modelli sono in grado di comprendere il contenuto testuale con maggiore profondità rispetto ai metodi tradizionali, rendendo possibile l'estrazione automatica di informazioni più ricche e accurate da pagine web complesse. Tuttavia, nonostante il potenziale di questa combinazione, la sua applicazione pratica presenta una serie di sfide tecniche.

Diversi progetti hanno esplorato l'uso di LLMs per migliorare i processi di crawling e scraping dei dati web, con l'obiettivo di rendere l'estrazione delle informazioni più precisa e affidabile. Tuttavia, molti di questi approcci si concentrano principalmente su specifiche applicazioni, come il recupero di informazioni da contenuti testuali complessi o l'estrazione di concetti tramite l'analisi del linguaggio naturale, trascurando alcuni aspetti rilevanti per i dati strutturati o semi-strutturati.

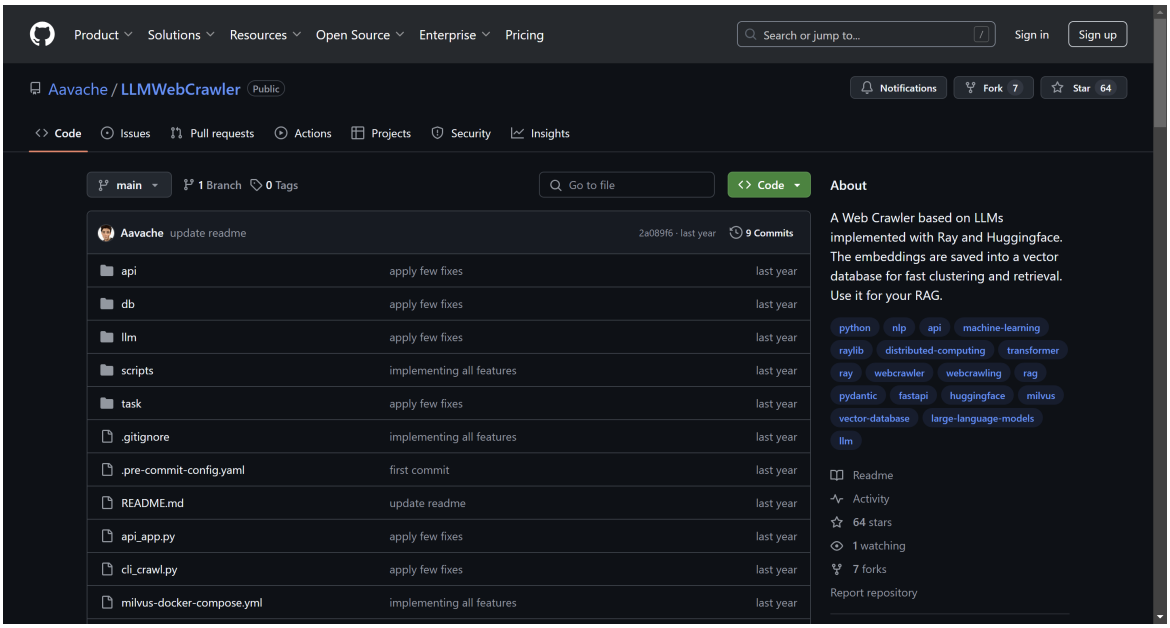

Figura 2.1 Repository GitHub del progetto LLMWebCrawler

In questa sezione vengono presentati due progetti open-source che integrano LLMs nei loro sistemi di crawling e analisi dei dati, illustrandone le funzionalità e le limitazioni.

2.1.1 LLMWebCrawler

Uno dei progetti più rilevanti in questo campo è il *LLMWebCrawler*, sviluppato e pubblicato su GitHub [1]. Questo sistema utilizza LLMs per analizzare le pagine web e per estrarre informazioni rilevanti, come parole chiave, concetti e collegamenti tematici. Il *LLMWebCrawler* è progettato per sfruttare le capacità di un modello di linguaggio per comprendere meglio il contesto dei contenuti web, potenziando così il recupero delle informazioni.

Uno degli aspetti distintivi di questo progetto è l'utilizzo di un *database vettoriale* per memorizzare i dati estratti. I database vettoriali sono ottimizzati per rappresentare e gestire i dati in forma di vettori numerici, consentendo operazioni rapide di similitudine tra vettori ed embedding. Questo approccio permette di eseguire ricerche semantiche e confronti tra documenti basati sul loro contenuto concettuale. Nonostante i vantaggi offerti dall'uso di LLM e di un database vettoriale, il *LLMWebCrawler* presenta alcune limitazioni significative. In particolare, il database vettoriale non è ideale per l'archiviazione di dati strutturati e semi-strutturati, come tabelle o informazioni specifiche presenti in documenti HTML, che sono comunemente analizzati durante il web scraping. Per il web scraping, infatti, i database relazionali o NoSQL risultano più adatti, poiché consentono di gestire in modo più efficiente dati organizzati in strutture più rigide e complesse. Nel nostro caso, dove l'obiettivo è

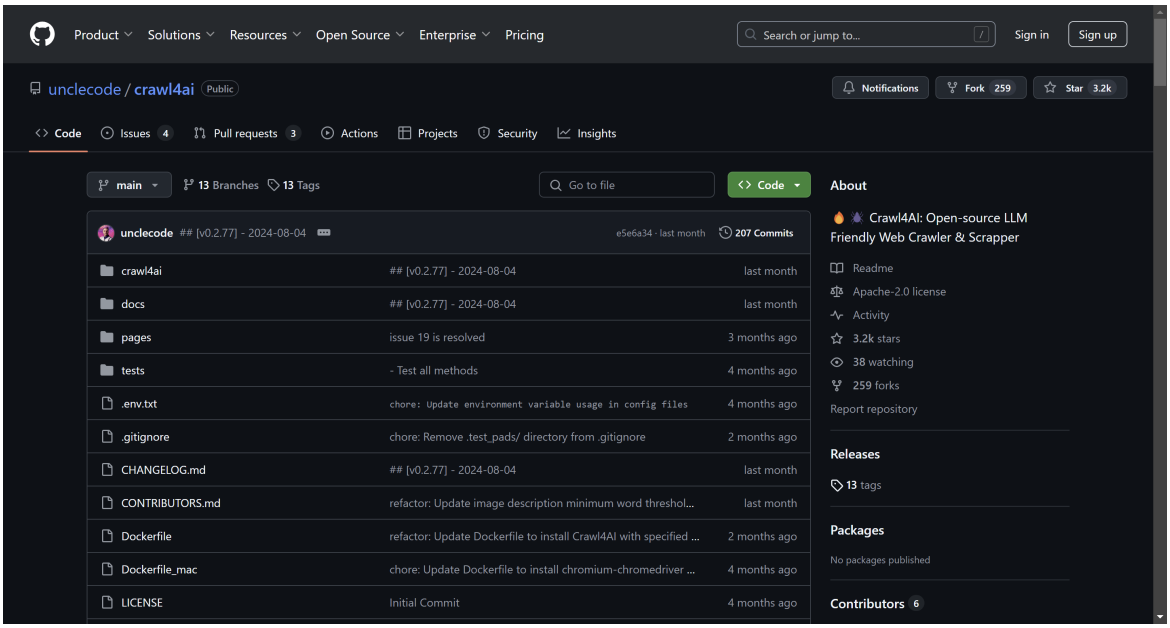

Figura 2.2 Repository GitHub del progetto Crawl4AI

estrarre e strutturare informazioni come titoli di eventi, date, luoghi e descrizioni, un database vettoriale non offre un vantaggio significativo.

2.1.2 Crawl4AI

Un altro progetto rilevante è *Crawl4AI*, un sistema di web crawling che mira a supportare applicazioni di intelligenza artificiale [3]. Questo crawler è progettato per raccogliere grandi quantità di dati da svariate fonti web e preparare tali dati per essere utilizzati in modelli di machine learning, compresi gli LLMs. Il progetto utilizza una combinazione di tecniche di scraping tradizionali e modelli di apprendimento per analizzare i dati raccolti, cercando di ottimizzare la qualità e la quantità delle informazioni ottenute.

Tuttavia, uno dei principali limiti del *Crawl4AI* è la sua incapacità di esplorare i dati in profondità. Sebbene sia in grado di raccogliere dati superficiali da un gran numero di pagine web, il sistema non è progettato per gestire l'analisi di contenuti più dettagliati o complessi, come quelli che si trovano nelle sottopagine o nei link interni di un sito web. Questo rappresenta un problema nel contesto della raccolta di dati per eventi, in cui le informazioni spesso si trovano in pagine annidate o richiedono una navigazione approfondita attraverso i link del sito. Inoltre, *Crawl4AI* si concentra prevalentemente sulla quantità dei dati piuttosto che sulla loro qualità. Manca di strumenti per garantire la precisione dell'estrazione delle informazioni, e il modello di LLM utilizzato non è ottimizzato per rispondere a domande

specifiche o estrarre dati in formati standardizzati come richiesto per la costruzione di un database di eventi.

2.2 Un sistema innovativo di Web Crawling con LLM

Il sistema proposto in questa tesi si distingue nettamente dai lavori esaminati sopra, introducendo un approccio più mirato e innovativo per la costruzione di un database di eventi per la città di Milano. L'obiettivo principale è quello di estrarre informazioni in modo preciso, efficiente e strutturato da una varietà di fonti web, utilizzando un modello LLM per migliorare l'accuratezza dell'estrazione e assicurare che le informazioni siano raccolte in maniera precisa.

A differenza di *LLMWebCrawler*, che memorizza i dati in vettori numerici per operazioni di similitudine, il sistema proposto utilizza tecniche di scraping tradizionali per estrarre i dati direttamente dal codice HTML delle pagine web. Questo approccio consente di raccogliere e strutturare dati come il titolo dell'evento, la data, il luogo e la descrizione, salvando tutto in formato JSON. Questo formato, molto più adatto per dati strutturati e semi-strutturati, facilita anche la loro gestione in database relazionali o NoSQL, rendendolo ideale per il tipo di informazioni che si intende raccogliere.

Un'altra innovazione del sistema è la capacità di esplorare le pagine web in profondità. Utilizzando i link presenti sui siti di eventi, il crawler è in grado di navigare attraverso sottopagine e trovare informazioni che non sono immediatamente accessibili. Questo permette al sistema di raccogliere dati più completi e accurati rispetto a sistemi come *Crawl4AI*, che si limitano a raccogliere informazioni superficiali.

Inoltre, il sistema è stato progettato per essere facilmente accessibile e aggiornabile. Il codice sarà scritto su una piattaforma Google Colab, rendendolo semplice da modificare e da eseguire. Questo consente di ripetere lo scraping dei dati più volte per garantire che i dati siano sempre aggiornati con le ultime informazioni sugli eventi. Grazie a questa flessibilità, il sistema può essere continuamente migliorato e adattato alle nuove esigenze.

In conclusione, il sistema descritto in questa tesi rappresenta un significativo avanzamento nello stato dell'arte, poiché combina le tecniche più efficaci di crawling e analisi dei dati con le capacità avanzate dei modelli LLM, creando una soluzione innovativa per l'estrazione e la gestione delle informazioni sugli eventi.

3

Tecnologie abilitanti

3.1 Tecniche di Web Crawling

In questa sezione vengono presentati i concetti e le applicazioni principali del web crawling. Viene analizzata l'architettura di un web crawler e discusso l'utilizzo di questa tecnologia, in particolare della fase di web scraping, per l'estrazione di informazioni pertinenti dalle pagine web relative agli eventi.

3.1.1 Concetti di base

Un web crawler, noto anche come spider o bot, è un software progettato per navigare e raccogliere informazioni dalle pagine web. I web crawler sono strumenti essenziali nel mondo dell'informatica e dell'analisi dei dati, utilizzati per automatizzare la raccolta e memorizzazione di dati utili per vari scopi, tra cui l'indicizzazione dei contenuti per i motori di ricerca, la raccolta di dati per analisi di mercato e la costruzione di database.

Uno degli utilizzi più noti dei web crawler è l'indicizzazione dei contenuti web per i motori di ricerca come Google, Bing e Yahoo. I crawler esplorano il web, analizzano i contenuti delle pagine e aggiornano gli indici dei browser. Questo processo rende possibile per gli utenti trovare informazioni rilevanti attraverso le query di ricerca.

I web crawler vengono inoltre utilizzati per raccogliere dati da varie fonti online per effettuare analisi competitive, monitoraggio dei prezzi, raccolta di recensioni di prodotti

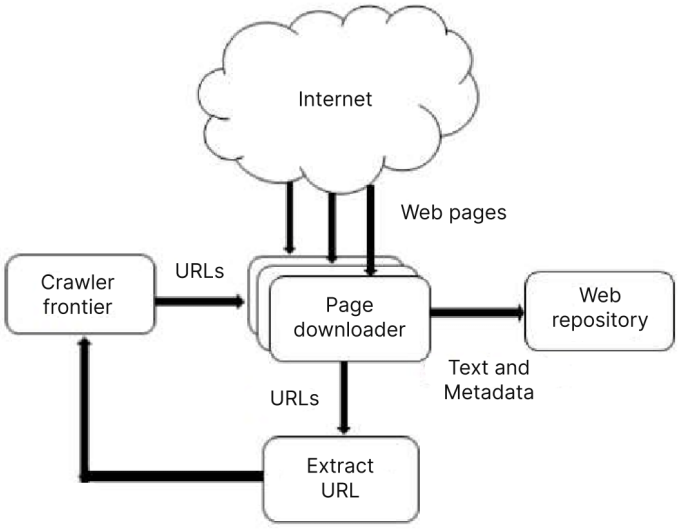

Figura 3.1 Funzionamento di un Web Crawler

e analisi delle tendenze di mercato. Ad esempio, un'azienda può monitorare i prezzi dei prodotti sui siti web dei concorrenti per ottimizzare la propria strategia di prezzo.

In ambito finanziario i crawler servono per l'analisi dei rischi e la valutazione di potenziali investimenti, mentre in ambiti legali vengono impiegati per rilevare violazioni della proprietà intellettuale e monitorare contenuti piratati.

L'utilizzo dei web crawler deve essere effettuato rispettando le normative legali ed etiche. Alcuni siti web utilizzano il file robots.txt per comunicare ai crawler quali pagine possono essere visitate e quali no. È fondamentale rispettare queste indicazioni per evitare problemi legali e tecnici. Inoltre, i crawler devono essere progettati per evitare di sovraccaricare i server web, rispettando i limiti di frequenza delle richieste.

In conclusione, i web crawler sono strumenti potenti e versatili che giocano un ruolo cruciale nella raccolta e nell'analisi dei dati online. La loro applicazione varia dall'indicizzazione dei motori di ricerca alla creazione di database tematici, offrendo un'ampia gamma di possibilità per migliorare l'accesso alle informazioni e supportare decisioni strategiche.

3.1.2 Architettura e funzionamento di un Web Crawler

Il funzionamento di un web crawler può essere suddiviso in diverse fasi chiave:

1. **Inizializzazione con URL Seed**: Alla base dell'architettura di un web crawler c'è il *modulo di avvio*, che utilizza i seed URLs, ovvero gli URL delle pagine web in analisi, come punto di partenza per l'intero processo di crawling. Il modulo di avvio invia richieste HTTP ai server web corrispondenti per scaricare il contenuto delle pagine

web. Una volta ricevuta la risposta dal server, il contenuto della pagina viene passato al modulo di parsing.

2. **Parsing del Contenuto**: Il *modulo di parsing* è responsabile dell'analisi del codice HTML delle pagine scaricate. Durante questa fase, il parser estrae i link ad altre pagine web presenti nel codice HTML. Questi link vengono quindi aggiunti a una coda di URL da visitare, nota come coda di crawling. Il modulo di parsing non si limita solo a estrarre i link, ma può anche identificare e raccogliere dal codice HTML informazioni specifiche richieste dall'utente.

3. **Web Scraping**: Durante la fase di parsing, il *modulo di web scraping* si occupa di estrarre dati specifici dalle pagine HTML scaricate, utilizzando tecniche automatizzate per individuare e raccogliere le informazioni desiderate. Ad esempio, se l'obiettivo è raccogliere dettagli sugli eventi, il modulo di scraping può analizzare il codice HTML alla ricerca di elementi specifici come titoli, date e luoghi all'interno di determinati tag HTML. L'estrazione può essere raffinata ulteriormente utilizzando espressioni regolari o tecniche di pulizia dei dati, per garantire che solo le informazioni pertinenti vengano raccolte.

4. **Scoperta dei Link**: La *coda di crawling* è un componente importante dell'architettura. Gestisce la lista degli URL che il crawler deve visitare. È fondamentale che la coda sia gestita in modo efficiente per evitare visite ripetute e garantire che il crawler esplori le pagine in modo ordinato. Algoritmi di gestione della coda possono implementare priorità per determinati URL, basandosi su criteri come la frequenza di aggiornamento del contenuto o l'importanza del sito web.

5. **Filtraggio e Gestione dei Dati**: Durante il crawling, i dati raccolti possono essere filtrati in base a criteri specifici, come la rimozione di duplicati o l'estrazione di informazioni pertinenti. Il crawler può anche ignorare determinati tipi di contenuti in base a regole predefinite.

6. **Salvataggio dei Dati**: Le informazioni estratte vengono memorizzate in un database o in un'altra forma di archiviazione strutturata. Questo consente un facile accesso e analisi dei dati raccolti.

L'architettura di un web crawler è un insieme complesso di componenti interconnessi ed è progettata con attenzione per garantire efficienza, scalabilità e robustezza.

La scalabilità è un aspetto chiave dell'architettura di un web crawler. Per gestire un numero crescente di pagine web, al sistema conviene distribuire il carico su più macchine.

Questo può essere ottenuto implementando un'architettura distribuita, dove più istanze del crawler lavorano in parallelo, coordinandosi attraverso il modulo di coordinamento e condividendo la coda di crawling e il database.

Infine, l'architettura deve includere meccanismi di gestione degli errori e di recupero. Poiché il web è un ambiente dinamico e imprevedibile, il crawler deve essere in grado di gestire situazioni come timeout delle richieste, errori di rete e modifiche ai siti web. I moduli di gestione degli errori rilevano e registrano questi problemi, tentando di ripetere le operazioni fallite o di saltare i link problematici.

Web Scraping per l'estrazione di informazioni dai siti web degli eventi

Il web scraping è la tecnica utilizzata per estrarre dati in modo automatico da siti web. Si tratta di un processo attraverso il quale le informazioni presenti in una pagina web, solitamente strutturate in formato HTML, vengono trasformate in dati utilizzabili e manipolabili per vari scopi. Questo processo è molto utilizzato in applicazioni come l'analisi di mercato, il monitoraggio dei prezzi, la raccolta di recensioni di prodotti, e molto altro ancora.

Il web scraping è un sottoprocesso del web crawling, ovvero, come spiegato in precedenza, il processo di esplorazione automatica del web per scoprire e scaricare pagine web a partire da un insieme di URL di base (seed URLs). I web crawler navigano tra i link per identificare e raccogliere pagine web, che poi possono essere analizzate e scansionate per l'estrazione dei dati durante la fase di web scraping.

Una volta che il crawler ha individuato e scaricato una pagina, entra in gioco lo scraping, che si occupa di analizzare la struttura della pagina per estrarre i dati desiderati. Ad esempio, un web crawler può essere utilizzato per raccogliere tutte le pagine di un sito web di eventi, e successivamente, il web scraping può estrarre informazioni specifiche come titolo, luogo, durata degli eventi.

Queste tecniche sono fondamentali per automatizzare la raccolta e l'analisi dei dati su larga scala e sono ampiamente utilizzate in settori che necessitano di grandi quantità di dati, come il marketing, la ricerca e l'intelligenza artificiale.

Il web scraping è spesso realizzato tramite l'uso di script o software dedicati che simulano l'interazione di un utente con una pagina web, navigando attraverso i contenuti e raccogliendo le informazioni necessarie. Python è tra i linguaggi di programmazione più usati, grazie a diverse librerie semplifica il processo di estrazione dei dati.

Utilizzando Python e la libreria *BeautifulSoup*, è possibile automatizzare questo processo, rendendolo efficiente e scalabile in pochi passaggi:

1. **Scaricamento delle Pagine Web**: Utilizzando Python, il contenuto delle pagine web può essere scaricato tramite librerie come *requests*. Questa libreria invia una richiesta HTTP al sito web e riceve il codice HTML della pagina, che verrà poi analizzato.

```
import requests

url = "https://www.sitoeventi.com/eventi"
response = requests.get(url)

# Verifica dello stato della richiesta
if response.status_code == 200:
    pagina_html = response.text
else:
    print("Errore nel recupero della pagina")
```

Listing 3.1 Codice Python per il recupero della pagina HTML di un sito

In questo esempio, la pagina web viene scaricata e il contenuto HTML viene memorizzato nella variabile *pagina-html*.

2. **Parsing del Contenuto con BeautifulSoup**: Una volta ottenuto il codice HTML della pagina, si utilizza *BeautifulSoup* per analizzarlo ed estrarre i dati desiderati. *BeautifulSoup* è una potente libreria di parsing HTML che permette di navigare nella struttura delle pagine e identificare gli elementi rilevanti.

```
from bs4 import BeautifulSoup

# Creazione di un oggetto BeautifulSoup
soup = BeautifulSoup(pagina_html, 'html.parser')

# Estrazione dei dati sugli eventi
eventi = soup.find_all('div', class_='evento')

for evento in eventi:
    titolo = evento.find('h2').text
    data = evento.find('span', class_='data').text
    luogo = evento.find('span', class_='luogo').text
    descrizione = evento.find('p', class_='descrizione').text

    print(f"Titolo: {titolo}\nData: {data}\nLuogo: {luogo}\n
    Descrizione: {descrizione}\n")
```

Listing 3.2 Codice Python per il parsing del contenuto della pagina HTML

In questo esempio, utilizziamo BeautifulSoup per trovare tutti gli elementi HTML che contengono informazioni sugli eventi. In particolare, stiamo cercando elementi *div* con la classe *evento*, e da lì estraiamo il titolo, la data, il luogo e la descrizione.

3. **Salvataggio dei Dati in un file**: Una volta puliti e normalizzati, i dati sugli eventi possono essere memorizzati in un file.

L'utilizzo di Python e BeautifulSoup per il web scraping offre numerosi vantaggi. Innanzitutto, permette di automatizzare l'intero processo di scraping, riducendo significativamente la necessità di intervento manuale. Questo consente di risparmiare tempo e risorse, soprattutto quando si devono gestire grandi quantità di dati. La flessibilità di BeautifulSoup è un altro punto di forza, poiché consente di adattare facilmente gli script a una vasta gamma di strutture HTML, indipendentemente dalla complessità o dalla qualità del codice sorgente della pagina web. Inoltre, grazie alla potenza di Python, è possibile scalare l'intero processo per gestire un numero elevato di fonti e dati, rendendolo adatto anche per progetti di grandi dimensioni. Infine, Python consente l'implementazione di tecniche avanzate di gestione degli errori e ottimizzazione delle performance, garantendo così l'efficienza del processo di scraping anche in ambienti dinamici e complessi.

3.2 Large Language Models (LLMs)

Questa sezione introduce i Large Language Models (LLMs) e la loro utilità nell'estrazione dei dati in combinazione con un web crawler. Verrà studiata l'implementazione e l'addestramento di un LLM e le potenzialità di utilizzo di questa tecnologia per migliorare l'accuratezza e la completezza delle informazioni estratte sugli eventi.

3.2.1 Introduzione ai LLMs e principali applicazioni

I Large Language Models (LLMs) rappresentano una delle tecnologie più rivoluzionarie nel campo dell'elaborazione del linguaggio naturale (NLP). Addestrati su enormi quantità di dati testuali, gli LLMs sono modelli computazionali in grado di processare il linguaggio umano in modo profondo e generare risposte che appaiono straordinariamente naturali e contestuali.

LLM più grandi e capaci sono reti neurali artificiali costruite con un'architettura basata su un trasformatore-decodificatore , che consente un'elaborazione e una generazione efficiente di dati di testo su larga scala, addestrate usando l'apprendimento autosupervisionato o l'apprendimento semisupervisionato. Miglioramenti nell'architettura e l'incremento delle

risorse computazionali hanno permesso di creare modelli con miliardi di parametri, capaci di affrontare compiti linguistici sempre più complessi.

Tra le principali applicazioni degli LLMs, si possono identificare le seguenti:

- **Risposta automatica a domande**: Uno degli utilizzi più comuni e diffusi degli LLMs è la loro capacità di fornire risposte a domande varie e complesse. Grazie alla loro comprensione del contesto e della semantica, i modelli possono fornire risposte dettagliate e articolate, che vanno oltre la semplice ricerca di parole chiave.

- **Riepilogo di testi**: La generazione di riassunti accurati è una delle funzioni chiave degli LLMs, soprattutto per l'analisi di documenti complessi. Modelli come GPT-4 sono in grado di identificare i punti salienti di un testo e di presentarli in modo conciso, riducendo significativamente il tempo necessario per estrarre le informazioni principali.

- **Generazione di contenuti**: La generazione automatica di testi è un'area in cui gli LLMs hanno dimostrato grande potenzialità. Questi modelli possono produrre articoli, descrizioni di prodotti e report.

- **Traduzione automatica**: La traduzione linguistica è un'altra area in cui gli LLMs hanno portato a un notevole miglioramento. Rispetto ai traduttori automatici tradizionali, modelli come GPT-4 sono in grado di tradurre non solo parole, ma anche sfumature e contesto culturale. Questo riduce gli errori tipici delle traduzioni automatiche precedenti, che spesso ignoravano il contesto, portando a traduzioni meccaniche e imprecise.

- **Supporto per chatbot**: Gli LLMs sono spesso utilizzati come motori dietro chatbot avanzati per il servizio clienti.

Nel contesto dell'estrazione di informazioni, gli LLMs stanno rivoluzionando il modo in cui i dati non strutturati vengono elaborati e interpretati. L'estrazione di informazioni da pagine web, articoli di giornale o altre risorse testuali spesso richiede di individuare dettagli precisi, come date, luoghi, persone o eventi per risultare efficace.

In sintesi, i Large Language Models rappresentano una delle tecnologie più promettenti per l'elaborazione del linguaggio naturale, grazie alla loro capacità di generalizzare il linguaggio e di applicarsi in una vasta gamma di contesti. Le loro applicazioni spaziano dalla risposta automatica alle domande alla generazione di contenuti, dalla traduzione alla sintesi dei testi, fino al supporto per chatbot avanzati e l'estrazione di informazioni dai dati non strutturati. Questo rende gli LLMs strumenti potenti e versatili per affrontare le sfide della moderna gestione delle informazioni.

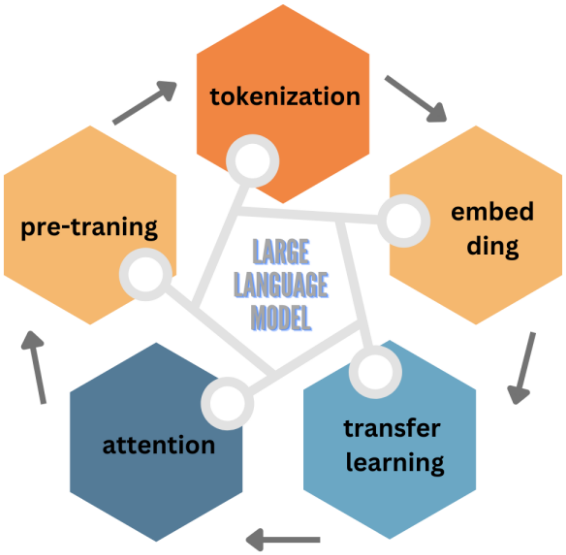

Figura 3.2 Elementi costitutivi di un LLM

3.2.2 Implementazione e addestramento di un LLM

L'implementazione di un Large Language Model (LLM) è un processo altamente sofisticato che richiede una combinazione di dati di grande volume, potenza computazionale e tecniche avanzate di ottimizzazione. I LLM sono addestrati su enormi quantità di testi provenienti da una varietà di fonti, come libri, articoli e pagine web. Questo vasto corpus di dati consente ai modelli di apprendere e generalizzare il linguaggio, permettendo loro di fornire risposte pertinenti e accurate a nuove domande o situazioni.

Il cuore della tecnologia degli LLM è costituito da architetture basate su reti neurali profonde, in particolare i *Transformers*. I Transformers hanno rivoluzionato l'approccio all'elaborazione del linguaggio naturale grazie alla loro capacità unica di gestire e analizzare sequenze di testo in parallelo. Questo è un passo avanti significativo rispetto ai modelli precedenti, che elaboravano il testo in modo sequenziale, limitando la loro capacità di comprendere il contesto completo di una frase o di un documento.

L'architettura a Transformers utilizza due principali componenti: l'*Encoder* e il *Decoder*. L'Encoder è responsabile dell'analisi del testo in ingresso e della creazione di una rappresentazione interna che cattura il significato e il contesto delle parole, un processo reso possibile anche grazie alla tokenizzazione e all'uso di *word embeddings*, che convertono i testi in numeri, rappresentando le parole in uno spazio vettoriale continuo, dove le parole semanticamente simili sono vicine tra loro, migliorando così la comprensione semantica del testo da parte del modello. Il Decoder utilizza queste rappresentazioni per generare il testo

di output, che può essere una risposta a una domanda, un riassunto di un documento o una traduzione in un'altra lingua.

Un aspetto fondamentale nel funzionamento dei Transformers riguarda i *parametri* del modello, che includono principalmente i pesi e i bias delle connessioni tra i nodi della rete neurale. Durante l'addestramento, il modello aggiorna questi parametri in base ai dati di input e alle previsioni, con lo scopo di migliorare le prestazioni del modello stesso. I pesi determinano l'importanza di ciascuna connessione tra i nodi, mentre i bias permettono al modello di effettuare aggiustamenti più precisi. Il numero di parametri può essere enorme nei modelli di grandi dimensioni, influenzando la loro capacità di apprendere e generalizzare.

Un'altra fase fondamentale nell'addestramento degli LLM è il *pre-training*, dove il modello viene inizialmente addestrato su una vasta gamma di dati per apprendere una rappresentazione generale del linguaggio. Successivamente, il modello può essere ulteriormente affinato tramite il *fine-tuning* su dataset più specifici per migliorare le sue prestazioni in compiti particolari.

Per addestrare un LLM, è necessario un grande potere computazionale. Durante l'addestramento, il modello apprende a predire la parola successiva in una sequenza di testo basandosi sulle parole precedenti. Questo processo, chiamato *apprendimento supervisionato*, aiuta il modello a costruire una comprensione sempre più sofisticata del linguaggio.

Infine, l'ottimizzazione del modello è una fase cruciale, che implica la regolazione dei parametri per migliorare la precisione delle previsioni. Gli algoritmi di ottimizzazione sono utilizzati per aggiornare i pesi della rete neurale in modo efficiente durante l'addestramento.

3.2.3 GPT-4o e GPT-4o mini

Attualmente uno degli esempi più avanzati di LLM è il modello **GPT-4o**, una versione ottimizzata di GPT-4, un modello di intelligenza artificiale avanzato basato su transformer sviluppato da OpenAI. Questo modello, rilasciato il 13 maggio 2024, è stato progettato per offrire prestazioni superiori, mantenendo al contempo un elevato grado di efficienza. Uno degli obiettivi principali di GPT-4o è ridurre il consumo di risorse computazionali e migliorare la velocità di elaborazione, consentendo l'integrazione di modelli di linguaggio avanzati anche in contesti in cui la potenza di calcolo è limitata. Questi miglioramenti sono stati ottenuti senza sacrificare la qualità delle risposte generate, che rimangono altamente accurate e pertinenti.

GPT-4o introduce una maggiore capacità di comprendere il contesto linguistico, il che si traduce in una migliore interpretazione di testi complessi. Questa maggiore precisione nel riconoscere e gestire il contesto riduce significativamente gli errori comuni legati all'am-

biguità del linguaggio naturale, rendendo le risposte non solo più accurate, ma anche più coerenti e fluide.

Il modello **GPT-4o mini** rappresenta una versione ridotta e ottimizzata di GPT-4o, progettata per ambienti con risorse computazionali limitate. Sebbene sia più compatto, GPT-4o mini conserva una notevole capacità di comprensione e generazione del linguaggio, risultando particolarmente adatto per applicazioni in tempo reale o per scenari in cui è necessario bilanciare efficienza e prestazioni.

Nonostante la sua dimensione ridotta, GPT-4o mini mantiene molte delle funzionalità avanzate della versione completa, come l'interpretazione contestuale e la capacità di generalizzare risposte su una vasta gamma di argomenti.

Model	**Description**	**Context Window**	**Max Output Tokens**
`gpt-4o`	GPT-4o: Our high-intelligence flagship model for complex, multi-step tasks. GPT-4o is cheaper and faster than GPT-4 Turbo.	128,000 tokens	4,096 tokens
`gpt-4o-mini`	GPT-4o-mini: Our affordable and intelligent small model for fast, lightweight tasks. GPT-4o mini is cheaper and more capable than GPT-3.5 Turbo. Currently points to gpt-4o-mini-2024-07-18.	128,000 tokens	16,384 tokens

Tabella 3.1 Comparazione dei modelli GPT-4o e GPT-4o-mini. [2]

In tabella vengono confrontati i due modelli evidenziando le differenze principali in termini di capacità e utilizzi, confrontando i due parametri fondamentali:

- **Context Window**: Indica il numero massimo di token (unità linguistiche) che il modello può processare e mantenere in memoria durante una singola richiesta o interazione. Per entrambi i modelli, il valore è di 128.000 token. Ciò è particolarmente utile in compiti che richiedono una lunga memoria di contesto, come la comprensione di documenti estesi o conversazioni complesse.

- **Max Output Tokens**: Questo valore indica il numero massimo di token che il modello può generare in una singola risposta. Nel caso di GPT-4o, il limite massimo è di 4.096 token, mentre per GPT-4o-mini è molto più elevato, con un massimo di 16.384 token. Un numero maggiore di token di output consente al modello di produrre risposte più lunghe e dettagliate, il che può essere utile per compiti come la generazione di dati strutturati o la creazione di testi complessi.

In sintesi, entrambi i modelli condividono la stessa capacità di contesto, ma differiscono notevolmente nella quantità di testo che possono generare in uscita, con GPT-4o-mini che offre una maggiore flessibilità in termini di risposte più lunghe e dettagliate, rendendolo maggiormente adatto all'analisi ed estrazione di dati.

3.2.4 Analisi delle potenzialità di utilizzo di LLMs per l'estrazione di informazioni sugli eventi

L'utilizzo degli Large Language Models (LLMs) per l'estrazione di informazioni da dati non strutturati, come quelli raccolti da un web crawler, rappresenta un significativo avanzamento tecnologico che può facilitare il modo in cui analizziamo i dati. Gli LLMs offrono numerose potenzialità per migliorare la precisione e la rilevanza delle informazioni che vogliamo raccogliere sugli eventi. Sfruttando le loro capacità avanzate di comprensione del linguaggio e di analisi contestuale si evidenziano numerosi benefici tra cui:

- **Riconoscere eventi rilevanti**: Gli LLMs, grazie alla loro capacità di comprendere il contesto e di interpretare il linguaggio in modo profondo, possono identificare eventi di interesse anche all'interno di testi complessi e ambigui. Questo significa che possono estrarre informazioni pertinenti da una vasta gamma di fonti, inclusi articoli di notizie, post sui social media, report aziendali e documenti accademici. I modelli sono in grado di riconoscere eventi significativi anche quando le descrizioni sono parziali, vaghe o incorporate in narrazioni più ampie. Al contrario di semplici script di codice, essi possono identificare eventi e segnalarli come rilevanti.

- **Estrarre dati chiave**: Gli LLMs sono particolarmente abili nel localizzare e estrarre dettagli cruciali come date, luoghi, persone coinvolte e altri elementi rilevanti che caratterizzano un evento, soprattutto quando questi dati sono sparsi nel testo oppure non hanno un formato standard che ne permette il diretto recupero. Questa capacità di identificare e catalogare informazioni specifiche migliora notevolmente l'accuratezza dell'estrazione dei dati, facilitando la creazione di database informativi e la generazione di report dettagliati. Ad esempio, un LLM potrebbe analizzare un articolo di notizie e estrarre automaticamente informazioni su una manifestazione, comprese le date di svolgimento, la località e i principali partecipanti, rendendo questi dati facilmente accessibili e utilizzabili per ulteriori elaborazioni.

- **Riassumere ed analizzare**: Gli LLMs non si limitano a estrarre testo, ma sono anche in grado di analizzare e rielaborare il risultato. Ad esempio un LLM può generare un riassunto dell'evento, permettendo una categorizzazione più ampia oltre ai dati

oggettivi raccolti, oppure può analizzare commenti sui social media o articoli di opinione per determinare se la reazione generale a un evento è positiva, negativa o neutra, per avere una panoramica di ciò che succederà. Questo tipo di analisi del sentiment può fornire ulteriori insights sul modo in cui gli eventi sono percepiti e influenzano l'opinione pubblica.

L'integrazione di LLMs nell'estrazione di informazioni sugli eventi non solo migliora l'efficienza del processo, automatizzando e accelerando la raccolta e l'elaborazione dei dati, ma apre anche nuove possibilità per applicazioni avanzate. Tra queste, il *monitoraggio in tempo reale* diventa particolarmente rilevante: gli LLMs possono essere utilizzati per tracciare e analizzare eventi in tempo reale, fornendo aggiornamenti tempestivi e informazioni dettagliate senza la necessità di intervento umano costante. Inoltre, le capacità predittive degli LLMs possono essere sfruttate per *analisi predittive*, dove i modelli analizzano i dati storici e le tendenze per anticipare futuri eventi o sviluppi, supportando decisioni strategiche in ambiti come il business, la sicurezza e la pianificazione.

In conclusione, l'adozione di LLMs per l'estrazione di informazioni sugli eventi rappresenta una trasformazione significativa nel modo in cui gestiamo e utilizziamo i dati non strutturati. La loro capacità di riconoscere eventi rilevanti, estrarre e raffinare dati chiave offre strumenti potenti per migliorare la precisione, l'affidabilità e l'efficacia delle informazioni sugli eventi, con benefici tangibili per molteplici settori e applicazioni.

Utilizzando un prompt specifico tramite API di GPT-4o mini, è possibile fornire al modello un testo o un contenuto da cui si desidera estrarre informazioni dettagliate come titolo, data, luogo degli eventi o qualsiasi altro tipo di dato rilevante. Il modello è in grado di analizzare il contenuto in modo accurato, comprendendo il contesto e identificando le informazioni chiave necessarie. Successivamente, può essere istruito per ordinare tali dati in un formato più organizzato, ad esempio generando un file contenente dati strutturati come tabelle, JSON, o CSV, che possono essere facilmente utilizzati in altri sistemi o applicazioni.

```
# Contenuto del file di input da passare a LLM
file_content = read_file_content('input.txt')

# Scrivo il prompt per il modello
system_content = """

Ti verra' inviato un file testo contente informazioni sugli eventi.
Devi generare un file contentente dati strutturati con questa
   struttura...

"""
```

```
# Creo il processo
completion = client.chat.completions.create(
    model="gpt-4o-mini",
    temperature=0,
    messages=[
        {"role": "system", "content": system_content},
        {"role": "user", "content": file_content}
    ],
    response_format={ "type": "json_object" }
)
```

Listing 3.3 Codice Python che esegue l'estrazione di informazioni tramite un Prompt al modello gpt-4o-mini

In questo esempio di codice viene letto il contenuto di un file di testo chiamato 'input.txt' e viene memorizzato in una variabile chiamata 'file content'. Successivamente, viene scritto un prompt da inviare al modello GPT-4o mini, che include istruzioni su come strutturare i dati relativi agli eventi contenuti nel file. Il prompt spiega che il modello dovrà generare un file con dati strutturati in un formato specifico. Infine, il processo viene avviato inviando al modello sia il messaggio di sistema con le istruzioni, sia il contenuto del file letto. L'output generato dal modello sarà in formato JSON e con una temperatura impostata a 0 per risposte più precise e coerenti.

4

Implementazione del codice

In questo capitolo viene illustrata in dettaglio la pipeline che ha portato alla mia implementazione di un sistema di web crawling che utilizza un Large Language Model (LLM) per estrarre e strutturare informazioni sugli eventi nella città di Milano. Il codice è stato concepito per essere facilmente accessibile, in quanto interamente realizzato all'interno di un ambiente Google Colab. Questo approccio non solo permette di eseguire lo scraping in modo ripetuto, ma garantisce anche un costante aggiornamento delle informazioni estratte, facilitando modifiche rapide e agevoli al codice stesso.

L'approccio adottato combina tecniche tradizionali di web scraping con l'utilizzo di un modello avanzato come GPT-4o mini, rendendo possibile l'automazione dell'intero processo. Di seguito verrà fornita una descrizione dettagliata di ciascun componente della pipeline e dei singoli passaggi, con esempi di codice che dimostrano come le informazioni vengano raccolte, elaborate e trasformate.

In questo capitolo si esamineranno anche le tecnologie di gestione e archiviazione dei dati. Saranno identificate le fonti di dati sugli eventi e si discuterà la costruzione del database grazie all'utilizzo delle tecnologie e gli strumenti analizzati nei capitoli precedenti.

4.1 Descrizione della pipeline

La pipeline si compone di diverse fasi, ciascuna progettata per affrontare specifici aspetti del problema di estrazione dei dati:

1. **Selezione delle pagine web sugli eventi**: Questa fase prevede la selezione dei siti web utilizzati come fonti di eventi, in particolare dei link alle pagine contenenti la lista degli eventi organizzati da ogni sito.

2. **Web Crawling**: Questa fase prevede la raccolta dei link alle pagine descrittive di ogni evento a partire dalle pagine web principali. Il crawler naviga attraverso il codice HTML della pagina, individuando e memorizzando i collegamenti pertinenti.

3. **Web Scraping**: Una volta raccolti i link, si procede all'estrazione dei dati dagli eventi, come il titolo, la data, la descrizione e altre informazioni correlate dal codice HTML.

4. **Salvataggio dei dati in file di testo**: I dati estratti nella fase precedente, non ancora strutturati, vengono salvati in un file di testo, che funge da input per il modello LLM.

5. **Analisi tramite LLM**: Il modello LLM viene poi utilizzato per trasformare il testo in JSON strutturato, garantendo una rappresentazione chiara e coerente delle informazioni.

6. **Creazione dei file JSON**: Infine, il JSON generato viene salvato in un file, pronto per essere utilizzato da applicazioni che richiedono dati strutturati.

La pipeline è stata progettata per essere scalabile e flessibile, permettendo l'estrazione e l'aggiornamento dei dati sugli eventi con facilità.

4.2 Tecnologie e strumenti utilizzati

Per lo sviluppo del sistema sono state utilizzate diverse tecnologie e librerie open-source che hanno facilitato la realizzazione della pipeline:

- **Python**: Linguaggio di programmazione utilizzato per lo sviluppo del sistema, essendo indicato per la sua versatilità nella manipolazione di dati.

- **Requests**: Libreria Python utilizzata per inviare richieste HTTP e ottenere il contenuto delle pagine web.

- **BeautifulSoup**: Libreria Python utilizzata per analizzare il codice HTML e semplificare l'estrazione dei dati.

- **GPT-4o-mini**: Modello di LLM utilizzato per strutturare le informazioni in un formato JSON standard.

- **Google Colab**: Ambiente di sviluppo cloud-based utilizzato per scrivere ed eseguire il codice. Rende il sistema facilmente accessibile e modificabile.
- **JSON**: Formato di dati utilizzato per memorizzare e strutturare le informazioni estratte in modo standardizzato.

4.2.1 Import necessari

Per far funzionare correttamente il codice, è fondamentale importare le librerie necessarie per l'estrazione e manipolazione dei dati:

```
import requests
from bs4 import BeautifulSoup
import json
from openai import OpenAI
```

Queste librerie permettono di eseguire richieste HTTP, analizzare e manipolare il codice HTML, lavorare con formati di dati strutturati in JSON e utilizzare i modelli di intelligenza artificiale di OpenAI.

4.3 Selezione delle pagine web sugli eventi

La prima fase di sviluppo del sistema è stata la scelta dei siti web da includere per l'estrazione degli eventi. Molti siti limitano l'accesso al loro contenuto attraverso tecniche di protezione contro bot e crawling o restrizioni indicate nei file `robots.txt`. Di conseguenza è stato necessario escludere molti siti che non permettevano il libero accesso ai dati, ma è stato comunque possibile selezionare diverse fonti che coprono una vasta gamma di eventi, tra cui:

- **Eventi teatrali**: Rappresentazioni, spettacoli e performance dal vivo nei seguenti teatri:
  - Teatro alla Scala
  - Piccolo Teatro di Milano
  - Teatro Elfo Puccini
  - Teatro Litta
- **Mostre nei musei**: Esposizioni d'arte e mostre temporanee nei seguenti musei:
  - MUDEC - Museo delle Culture
  - Pinacoteca di Brera

  - Pirelli HangarBicocca

- **Concerti e gare**: Eventi musicali e gare sportive con una fonte dedicata che raccoglie tutti i concerti programmati a Milano:
  - Autodromo Nazionale di Monza
  - Concerti a Milano

La diversità delle fonti garantisce un'ampia copertura degli eventi culturali, permettendo al sistema di offrire una panoramica completa delle attività disponibili in città.

4.4 Estrazione delle informazioni tramite web crawling e web scraping

La fase di raccolta delle informazioni rappresenta il cuore della pipeline. Attraverso tecniche di crawlign e scraping, il codice estrae i dettagli principali degli eventi direttamente dal codice HTML delle pagine web. Ecco un esempio di come è stato implementato il processo di estrazione:

```
# URL della pagina principale degli eventi
url_principale = 'https://pagina-eventi'

# Funzione per ottenere il contenuto di una pagina
def get_page_content(url):
    response = requests.get(url)
    response.raise_for_status()
    return response.content

# Funzione per estrarre tutti i link agli eventi dalla pagina
   principale
def get_event_links(url):
    content = get_page_content(url)
    soup = BeautifulSoup(content, 'html.parser')
    # Estrazione di tutti i link alle pagine degli eventi
    event_links = soup.find_all('a', class_='link')
    return [a['href'] for a in event_links]
    return event_links

# Funzione per estrarre le informazioni di un evento da una pagina
   dell'evento
def get_event_details(url):
```

```
    content = get_page_content(url)
    soup = BeautifulSoup(content, 'html.parser')

    # Estrazione del titolo, data e categoria
    info_div = soup.find_all('div', class_='info')

    # Estrazione della descrizione
    description = soup.find('div', class_='description')

    return {
        'url': url,
        'descrizione': description,
        'event_info': event_info,
    }
```

Listing 4.1 Codice Python per la scoperta dei link alle pagine degli eventi e l'estrazione delle informazioni

Il codice sopra illustrato inizia raccogliendo i link agli eventi dalla pagina principale del sito, utilizzando tecniche di web crawling per individuare i collegamenti pertinenti. Una volta ottenuti tutti i link degli eventi, il codice prosegue iterando su ciascun link per estrarre dettagli rilevanti identificando specifici elementi HTML, come il titolo dell'evento, la descrizione, le date di inizio e fine, la categoria e altre informazioni chiave, come il luogo e l'URL della pagina dell'evento. Questi dati vengono raccolti in modo semi-strutturato e salvati in un file di testo temporaneo, che funge da archivio intermedio.

Questo file di testo verrà utilizzato per un ulteriore trattamento, in cui le informazioni vengono passate a un modello di intelligenza artificiale (LLM) per essere riformattate e strutturate in modo preciso. Successivamente, i dati estratti saranno analizzati per assicurare che tutte le informazioni siano complete e accurate, facilitando così l'automazione del processo di gestione dei dati sugli eventi e il loro utilizzo futuro.

4.5 Salvataggio dei dati in formato JSON tramite LLM

Una volta che le informazioni sugli eventi sono state raccolte e salvate in formato testuale, il passaggio successivo prevede l'utilizzo di un LLM per analizzare il testo grezzo e trasformarlo in un formato strutturato (JSON). Di seguito, viene mostrato un esempio di come il modello GPT-4o-mini viene utilizzato per questa fase:

```
file_content = read_file_content('input.txt')

# Scrivo il prompt per il modello
```

```
system_content = """

Ti verra' inviato un file testo contente informazioni sugli eventi.
Devi generare un json con le info degli eventi estraendole dal file.
Il json deve avere questa struttura

    {
      "title": "string",
      "url": "string",
      "location": "string",
      "address": "string",
      "date_start": "string", (formato: YYYY-MM-DD)
      "date_end": "string", (formato: YYYY-MM-DD)
      "category": "string",
      "description": "string"
    }

"""

# Creo il processo
completion = client.chat.completions.create(
    model="gpt-4o-mini",
    temperature=0,
    messages=[
        {"role": "system", "content": system_content},
        {"role": "user", "content": file_content}
    ],
    response_format={ "type": "json_object" }
)

# Salvo la risposta JSON in un file
save_to_json(response_json, 'output.json')
```

Listing 4.2 Codice Python per la creazione di dati strutturati da parte del LLM

L'output del modello viene quindi salvato in formato JSON, fornendo un insieme strutturato di informazioni sugli eventi. JSON, acronimo di JavaScript Object Notation, è un formato di scambio dati leggero e leggibile sia per gli esseri umani che per le macchine. Composto da coppie chiave-valore, JSON rappresenta dati complessi in modo semplice e chiaro. Questo file JSON può poi essere utilizzato per popolare database o altri sistemi di gestione dati.

Per ogni sito analizzato, sono stati effettuati i passaggi della pipeline descritta in prece-

denza, assicurando che tutte le informazioni sugli eventi venissero estratte in modo accurato e sistematico in file separati per ciascun sito. I file in output generati da questa fase di estrazione contengono informazioni preziose sugli eventi, che verranno ulteriormente analizzate nel prossimo capitolo. In particolare, saranno condotti esperimenti per valutare l'accuratezza delle informazioni estratte e la loro utilità per la costruzione di un database di eventi. Questo approccio ci permetterà di validare l'efficacia del sistema sviluppato e di ottimizzarlo in base ai risultati ottenuti.

```
{
    "events": [
        {
            "title": "NAVATE Jean Tinguely",
            "url": "https://pirellihangarbicocca.org/mostra/jean-
               tinguely/",
            "location": "Pirelli HangarBicocca",
            "address": "Via Chiese, 2, 20126 Milano MI",
            "date_start": "2024-10-10",
            "date_end": "2025-02-02",
            "category": "Mostra Futura",
            "description": "Jean Tinguely, pioniere dell'arte
               cinetica, esplora il movimento e il suono attraverso
               macchine rumorose e sculture performative, in una
               retrospettiva che celebra il suo legame con Milano."
        },
        ...
    ]
}
```

Listing 4.3 File Json di esempio generato dal LLM

Repository GitHub: Il codice completo è disponibile nel repository GitHub al seguente link: https://github.com/avigani3/llm-crawler.git

5

Esperimenti e risultati

In questo capitolo vengono analizzati gli esperimenti condotti per verificare il corretto funzionamento del sistema, sviluppato durante il lavoro di tesi e descritto nel capitolo precedente. Dopo la scrittura del codice per la generazione dei file JSON con le informazioni estratte sugli eventi, la fase successiva è quella di valutazione dei dati. L'obiettivo degli esperimenti è garantire che i file JSON prodotti rispettino le specifiche di formato, che contengano tutte le informazioni necessarie e che le informazioni estratte siano accurate.

5.1 Valutazione delle metriche

Essendo l'output del Large Language Model (LLM) costituito esclusivamente da codice JSON, sono state scelte le seguenti metriche per la valutazione dei dati estratti:

- **Conformità**: questa metrica valuta che i valori nei campi del JSON rispettino il formato corretto. Ad esempio, le date vengono verificate utilizzando espressioni regolari (*regex*) per accertarsi che seguano il formato indicato (YYYY-MM-DD). Se anche solo un campo non rispetta il formato previsto, l'intero JSON viene considerato non conforme e quindi non valido per l'uso successivo. Questo assicura che il formato dei dati rimanga rigorosamente uniforme.
- **Completezza**: questa metrica verifica che tutti i campi obbligatori siano presenti nel file JSON e che la struttura generata sia conforme allo schema stabilito nella fase di scrittura del prompt al modello. Utilizzando strumenti come *JSON Schema*,

viene garantito che i campi obbligatori siano correttamente popolati. Se anche un singolo campo obbligatorio risulta mancante, l'intero JSON non viene convalidato, evidenziando la mancanza di dati essenziali per l'accuratezza complessiva del sistema.

- **Accuratezza**: questa metrica assicura che i dati estratti siano rappresentativi delle informazioni effettivamente presenti nelle pagine web. Si eseguono confronti a campione per verificare che i dati estratti corrispondano fedelmente a quelli pubblicati sui siti analizzati. L'accuratezza viene valutata tramite un "match esatto" (*exact match*) dei valori, escludendo però campi come la descrizione, che può variare sensibilmente ed essere meno oggettiva. Questo criterio di valutazione dei dati estratti garantisce che ogni valore dei campi nel file JSON corrisponda esattamente a quelli delle fonti originali. Ciò riduce al minimo il rischio di errori e ambiguità, assicurando che le informazioni siano affidabili e coerenti e consentendo di identificare rapidamente i problemi nell'estrazione.

La valutazione è stata implementata direttamente nel codice per testare i file JSON prodotti, permettendo la correzione del codice in base agli errori riscontrati.

5.2 Metodologia e misurazione delle metriche

Per ogni file JSON creato, vengono applicati una serie di controlli per verificare la conformità, la completezza e l'accuratezza dei dati. Di seguito vengono descritti i metodi utilizzati per valutarli in modo oggettivo secondo ciascuna metrica:

5.2.1 Conformità e completezza

La conformità e la completezza dei dati estratti sono state misurate utilizzando uno schema JSON predefinito. Questo schema funge da modello di riferimento e specifica il tipo di dato atteso per ciascun campo (ad esempio, stringhe per titoli e URL, date per le informazioni temporali) e i pattern di formato richiesti. Un esempio di pattern importante è quello per la data, che deve rispettare il formato standard (YYYY-MM-DD), e viene validato tramite espressioni regolari per assicurare che il valore inserito sia corretto.

Lo schema JSON predefinito utilizzato per la validazione non è stato creato arbitrariamente, ma è lo stesso schema fornito al modello di intelligenza artificiale (Large Language Model) in fase di generazione dei dati. Ciò significa che durante la creazione del file JSON, il modello ha seguito esattamente queste specifiche per formattare i dati, rendendo la validazione particolarmente efficace e coerente. Di conseguenza, ogni file JSON generato dal modello è soggetto a una verifica rigorosa per assicurare che rispetti pienamente lo schema previsto.

Di seguito viene mostrato un esempio di codice utilizzato per effettuare questa validazione:

```
# Definizione dello schema JSON
schema = {
    "items": {
        "type": "object",
        "properties": {
            "title": {"type": "string"},
            "url": {"type": "string"},
            "location": {"type": "string"},
            "address": {"type": "string"},
            "date_start": {
                "type": "string",
                "pattern": "^\d{4}-\d{2}-\d{2}$" # Formato YYYY-MM-DD
            },
            "date_end": {
                "type": "string",
                "pattern": "^\d{4}-\d{2}-\d{2}$" # Formato YYYY-MM-DD
            },
            "category": {"type": "string"},
            "description": {"type": "string"}
        },
        "required": ["title", "url", "location", "address", "
           date_start", "date_end", "category", "description"]
    }
}

# Esegui la validazione del file secondo lo schema json
valida_json("output.json", schema)
```

Listing 5.1 Codice Python per la validazione dello schema JSON

Questo codice verifica che tutti i campi richiesti degli eventi siano presenti nel file JSON e che i loro valori siano nel formato corretto. Qualsiasi discrepanza o omissione viene segnalata con un errore. Inoltre, lo schema JSON viene valutato per assicurare che la struttura sia conforme a quanto specificato, garantendo la presenza di tutti i campi obbligatori. Il file di output viene convalidato in base a queste metriche solo se lo schema non genera errori in fase di analisi, ciò permette di individuare tempestivamente eventuali difformità nei dati estratti. Lo schema JSON utilizzato per la validazione impone che ogni campo obbligatorio sia presente e che rispetti il formato specificato. Se anche solo un singolo campo di un oggetto all'interno del file JSON non corrisponde a quanto definito nello schema (ad esempio

se un campo è mancante o vuoto, o non rispetta il formato richiesto), l'intero file JSON non verrà considerato valido. Questo approccio rigoroso è utile per assicurarsi che non vengano introdotti dati errati o parziali nel sistema, e permette di rilevare subito eventuali errori di estrazione, impedendo che vengano salvati o utilizzati dati incompleti o malformati.

5.2.2 Accuratezza

Per valutare l'accuratezza del sistema, gli eventi estratti vengono confrontati con un insieme di eventi di riferimento generati manualmente. Questi eventi sono stati generati personalmente a partire dalle informazioni pubblicate sui siti web selezionati. Per ogni sito, sono stati raccolti dai 3-5 eventi, garantendo che questi rappresentino fedelmente i dati reali presenti online. Questa raccolta manuale funge da file *gold standard* o verità di riferimento, contro cui verificare l'affidabilità delle informazioni estratte automaticamente dal sistema.

Il codice seguente illustra il processo di confronto tra gli eventi estratti e quelli di riferimento:

```
# Lista di oggetti di esempio da cercare
eventi_da_confrontare = [
        {
            "title": "Mostra␣Futura␣NAVATE␣Yukinori␣Yanagi",
            "url": "https://pirellihangarbicocca.org/mostra/yukinori-
               yanagi/",
            "location": "Pirelli␣HangarBicocca",
            "address": "Via␣Chiese,␣2,␣20126␣Milano␣MI",
            "date_start": "2025-03-27",
            "date_end": "2025-07-27",
            "category": "Mostra␣Futura"
        },
        {
            "title": "Mostra␣Futura␣NAVATE␣Jean␣Tinguely",
            "url": "https://pirellihangarbicocca.org/mostra/jean-
               tinguely/",
            "location": "Pirelli␣HangarBicocca",
            "address": "Via␣Chiese,␣2,␣20126␣Milano␣MI",
            "date_start": "2024-10-10",
            "date_end": "2025-02-02",
            "category": "Mostra␣Futura"
        }
]

# File JSON di output da verificare
file_json = "output.json"
```

```
# Verifica se gli eventi sono presenti nel file di output
risultati = verifica_eventi(file_json, eventi_da_confrontare)

# Stampa i risultati
for titolo, risultato in risultati.items():
    print(f"L'evento ' {titolo}' e' {risultato}.")
```

Listing 5.2 Codice Python per la verifica dell'accuratezza del JSON

Questo codice confronta gli eventi presenti nel file JSON con una lista di eventi di riferimento predefiniti, verificando che i dati estratti siano accurati e completi rispetto a quelli reali. Per ogni evento, vengono esaminati campi fondamentali come il titolo, l'URL, il luogo, la data di inizio e di fine, e la categoria, assicurandosi che corrispondano esattamente a quelli attesi. Il confronto avviene utilizzando un metodo di *exact match*, che richiede una corrispondenza esatta tra i valori dei campi degli eventi estratti e quelli degli eventi di riferimento. Non viene considerata la descrizione, in quanto essa è soggetta a una variabilità molto alta tra i siti web e potrebbe non seguire uno schema rigido o oggettivo, rendendo complesso un confronto esatto su questo specifico campo.

Gli eventi nel campione di riferimento devono essere presenti come oggetti identici nel file JSON. Se un evento estratto dal sistema contiene anche solo una minima differenza rispetto ai dati di riferimento (ad esempio un luogo o una data errati), l'intero file JSON non viene convalidato. Questo approccio rigoroso è fondamentale poiché il sistema deve garantire che tutti gli eventi rispettino rigorosamente i dati forniti dai siti web di origine. Ogni discrepanza, che si tratti di valori non conformi o della mancanza di un evento, viene segnalata come errore, impedendo la validazione del file JSON.

L'obiettivo è assicurarsi che le informazioni estratte siano completamente accurate rispetto ai dati reali, senza margine di errore, al fine di mantenere un alto livello di qualità nei dati finali. Questo tipo di confronto per esatta corrispondenza serve a garantire che non vi siano differenze tra gli eventi estratti e quelli originali, assicurando che il file JSON finale sia pronto per l'uso senza problemi di integrità. Solo se tutti gli eventi in campione corrispondono esattamente a quelli di riferimento, il file viene considerato valido. Al contrario, se anche solo un evento risulta non conforme, l'intero file JSON non viene accettato, segnalando così un problema nell'estrazione dei dati. Questo processo di validazione assicura che i dati finali siano rappresentativi e corretti prima di procedere con le fasi di utilizzo.

Sebbene il campione possa fornire una buona indicazione della qualità complessiva dell'estrazione, non offre una garanzia totale sulla correttezza di tutti i dati raccolti. Poiché il confronto viene eseguito solo su un campione, esiste la possibilità che errori non rilevati

Figura 5.1 Analisi di un errore di accuratezza

possano essere presenti negli eventi non inclusi nel campione stesso. Pertanto, sebbene il sistema consenta di ottenere un preciso livello di accuratezza, non può garantire la correttezza di tutti gli eventi estratti.

5.3 Analisi dei risultati

Di seguito viene riportata una tabella con i risultati delle verifiche effettuate per ciascun sito web di eventi, valutando le tre metriche descritte: conformità, completezza e accuratezza. Per ciascun sito è stato determinato il numero totale di eventi estratti dal sistema (dati aggiornati a ottobre 2024). In totale, sono stati estratti oltre mille eventi, considerando tutti i file JSON generati.

La **conformità** è considerata positiva quando tutti i campi presenti nel file JSON rispettano i formati e le specifiche definite nello schema JSON. In particolare, ogni campo obbligatorio deve soddisfare le regole di validazione, come il formato delle date e il tipo di dato atteso. Se anche uno solo dei campi non rispetta tali requisiti, la conformità viene valutata negativamente.

La **completezza** è valutata positiva quando tutti i campi obbligatori del file JSON sono stati correttamente popolati secondo lo schema corretto e non ci sono omissioni di dati rilevanti. Questo significa che non solo devono essere presenti tutti i campi richiesti, ma devono anche contenere informazioni significative e pertinenti. Se manca anche solo un campo obbligatorio, la completezza è considerata negativa.

Infine, l'**accuratezza** è misurata attraverso il numero di eventi presenti nel campione di riferimento che sono risultati esatti nei file di output, senza errori nei campi rilevanti come titolo, data, luogo e categoria. Se un evento estratto è conforme ai dati di riferimento, viene

conteggiato come preciso; al contrario, se vi è anche una sola discrepanza, quell'evento non viene considerato accurato. La valutazione complessiva dell'accuratezza dipende quindi dal numero totale di eventi estratti correttamente rispetto al campione utilizzato per il confronto. Ad esempio, per il *Teatro alla Scala*, il campione di riferimento nel file *gold standard* era composto da 5 eventi, e tutti e 5 sono stati estratti correttamente dal sistema, con i dati presenti in modo esatto nel file di output. Per questo motivo, l'accuratezza per il Teatro alla Scala è risultata di 5/5, dimostrando che il sistema ha riportato in modo preciso ogni evento del campione.

Queste metriche forniscono un quadro chiaro della qualità e dell'affidabilità dei dati estratti, evidenziando eventuali aree di miglioramento nel sistema di raccolta e validazione delle informazioni.

Sito Web	**N. Eventi**	**Conformità**	**Completezza**	**Accuratezza**
Teatro alla Scala	142	Positivo	Positivo	5/5
Piccolo Teatro di Milano	53	Positivo	Positivo	5/5
Teatro Elfo Puccini	60	Positivo	Positivo	4/4
Teatro Litta	38	Positivo	Positivo	0/5
MUDEC - Museo delle Culture	4	Positivo	Positivo	3/3
Pinacoteca di Brera	10	Positivo	Positivo	3/3
Pirelli HangarBicocca	6	Positivo	Positivo	3/3
Autodromo Nazionale di Monza	5	Positivo	Positivo	3/3
Concerti a Milano	719	Positivo	Positivo	5/5

Tabella 5.1 Risultati degli esperimenti per ciascun sito web di eventi, aggiornati ad ottobre 2024

I risultati delle verifiche hanno dimostrato che il sistema sviluppato è in grado di estrarre correttamente le informazioni sugli eventi, rispettando i formati e le strutture previste. Tuttavia, è stato riscontrato un errore di accuratezza nella generazione dei dati relativi agli eventi del *Teatro Litta*. In particolare, non venivano correttamente estratti i campi *categoria* e *durata*. Questo problema era legato a un errore nella fase di web scraping, in cui i dati grezzi non venivano salvati nel file di testo in modo adeguato poiché non venivano selezionati correttamente gli elementi HTML da cui attingere ai dati. Di conseguenza, il LLM utilizzato per l'analisi e l'elaborazione dei dati non era in grado di estrarre queste informazioni, poiché non erano presenti nel file di origine. L'individuazione di questo errore ha permesso di correggere il codice, migliorando ulteriormente l'affidabilità e la precisione del sistema.

In conclusione, i test condotti hanno dimostrato l'efficacia e l'affidabilità del sistema di estrazione automatizzata dei dati, confermando la qualità dei risultati ottenuti. I dati presenti nei file JSON generati dal sistema sono stati validati con successo, risultando conformi alle metriche di conformità, completezza e accuratezza stabilite. Le informazioni estratte sono

risultate coerenti con quelle contenute nelle pagine web di riferimento, assicurando che gli eventi siano stati riportati con precisione e senza omissioni.

Anche se l'analisi offre un'indicazione utile della qualità dell'estrazione, non assicura l'accuratezza di tutti i dati raccolti, è possibile che vi siano errori non identificati negli eventi esclusi dal campione. Inoltre, considerando che alcuni siti hanno un numero limitato di eventi, potrebbe essere interessante esplorare ulteriori fonti o tipologie di eventi per arricchire il database e migliorare la varietà delle informazioni disponibili.

Per ogni sito web analizzato, il sistema è comunque stato in grado estrarre in modo completo e accurato tutti gli eventi presenti, assicurando che dettagli chiave come titolo, descrizione, data, luogo e categoria venissero rappresentati correttamente nel file di output. La pipeline ha dimostrato di essere robusta nell'automatizzare la raccolta delle informazioni e nel generare dati strutturati pronti per essere integrati in altre applicazioni. Questo conferma che il sistema può essere utilizzato con successo per mantenere aggiornate le informazioni sugli eventi di Milano e per semplificare la gestione di un vasto numero di dati in modo efficiente e sistematico.

6
Conclusioni

Nelle conclusioni vengono discussi i risultati ottenuti rispetto agli obiettivi prefissati e forniti suggerimenti per possibili sviluppi futuri e miglioramenti del sistema proposto. L'intento principale era quello di realizzare un sistema innovativo, capace di raccogliere informazioni in maniera accurata ed efficiente, per supportare analisi avanzate e lo sviluppo di applicazioni. Si riflette sull'impatto potenziale del lavoro svolto, in un contesto più ampio della gestione degli eventi a Milano e oltre.

6.1 Discussione dei risultati ottenuti rispetto agli obiettivi prefissati

Gli obiettivi iniziali sono stati pienamente raggiunti sotto diversi aspetti:

- **Raccolta di dati sugli eventi di Milano**: Il sistema è stato in grado di raccogliere informazioni aggiornate da diverse fonti online, creando un database ricco di eventi culturali, teatrali e musicali, dimostrando l'efficacia del web crawler implementato. Sono stati estratti dati chiave come titolo, data, luogo e, opzionalmente, descrizioni e altre informazioni aggiuntive sugli eventi in città.
- **Progettazione di un web crawler efficiente**: Il web crawler sviluppato è risultato essere robusto ed efficiente, capace di navigare e estrarre dati da fonti eterogenee, garan-

tendo aggiornamenti periodici e mantenendo una buona solidità. L'implementazione ha dimostrato inoltre la capacità di gestire grandi volumi di dati.

- **Integrazione con LLM**: L'integrazione con i Large Language Models ha permesso di migliorare l'accuratezza delle informazioni estratte, grazie ad un'elaborazione dettagliata dei dati testuali raccolti. Questo ha contribuito a rendere più precise le informazioni strutturate all'interno dei file generati.

- **Progettazione di un sistema di raccolta dati**: La scelta di strutturazione dei file JSON prodotti ha permesso di gestire i dati in modo accurato, mantenendo la flessibilità necessaria per futuri aggiornamenti e integrazioni. Il sistema permette di accedere facilmente alle informazioni e di aggiornare i dati con facilità.

- **Valutazione della qualità dei dati estratti**: Durante lo sviluppo del progetto sono stati definiti e implementati criteri di valutazione per monitorare la qualità e la completezza dei dati estratti. È stata inoltre verificata la conformità dei dati raccolti con uno schema ben definito, assicurando così una struttura uniforme e facilmente gestibile.

Durante lo sviluppo della tesi ho inoltre acquisito una serie di competenze tecniche e pratiche che mi hanno permesso di affrontare sfide complesse legate all'estrazione e alla gestione dei dati. In particolare, ho potuto approfondire la conoscenza del linguaggio Python, un linguaggio con cui avevo inizialmente poca familiarità. Durante lo sviluppo del progetto sono diventato più abile nell'utilizzo di questo linguaggio di programmazione, imparando a sfruttare le sue librerie e strumenti per creare un sistema di web crawling efficace, integrato tramite API con modelli LLM. Questo mi ha consentito di gestire in modo automatico la raccolta e l'elaborazione di dati testuali in grandi volumi.

Le competenze acquisite non solo si sono rivelate fondamentali per il completamento di questo lavoro, ma sono anche altamente trasferibili. L'integrazione di strumenti innovativi come i LLM mi ha fornito una comprensione approfondita di come utilizzare queste nuove tecnologie per l'elaborazione automatica di informazioni. Queste capacità possono essere applicate in vari contesti, dalla gestione di database alla creazione di sistemi di intelligenza artificiale per la raccolta e l'interpretazione dei dati, anche in altri settori. Il lavoro svolto rappresenta quindi una base solida per affrontare progetti futuri in ambiti simili o completamente nuovi.

6.2 Suggerimenti per possibili sviluppi futuri e miglioramenti del sistema proposto

Sebbene i risultati ottenuti siano stati soddisfacenti, ci sono diversi aspetti che potrebbero essere migliorati in future evoluzioni del sistema. In particolare, si potrebbe ampliare il progetto integrando fonti di dati aggiuntive oltre ai siti web tradizionali. Di seguito, alcune proposte per sviluppi futuri:

- **Integrazione con altre fonti di dati**: Per arricchire ulteriormente il database degli eventi, potrebbe essere utile estrarre dati da piattaforme social come Facebook, Instagram o Twitter, e da newsletter tematiche. Queste fonti contengono spesso informazioni rilevanti su eventi di piccola e media dimensione, non sempre pubblicizzati sui siti web ufficiali.
- **Utilizzo di LLM per analisi di immagini e post**: Un'altra possibilità, collegata al punto precedente, sarebbe quella di estendere l'uso dei LLM per l'analisi di immagini o post multimediali presenti in rete. Questo permetterebbe di ottenere informazioni anche da formati non testuali, ampliando le fonti dei dati raccolti.
- **Estrazione di campi aggiuntivi**: Oltre ai campi principali come titolo, data e luogo, sarebbe interessante estrarre ulteriori informazioni sugli eventi, come recensioni degli utenti o informazioni sulle prenotazioni. Questo arricchirebbe il set di dati e migliorerebbe le potenziali applicazioni del sistema, rendendolo più versatile e utile per altri compiti.
- **Ottimizzazione delle prestazioni**: Sebbene il sistema sia già efficiente, ci sono margini di miglioramento per quanto riguarda la gestione di maggiori volumi di dati, specialmente nell'accesso a siti web che potrebbero imporre limitazioni di traffico o blocchi geografici. L'uso di server proxy e VPN potrebbe essere implementato per consentire l'accesso continuo alle fonti di dati, migliorando la stabilità e la velocità del web crawler e aumentando le performance del sistema.

Il lavoro svolto in questa tesi ha confermato la validità del sistema proposto per la raccolta dei dati sugli eventi di Milano, dimostrando come l'integrazione di web crawling e modelli LLM possa risolvere efficacemente il problema della frammentazione delle informazioni. Il sistema è in grado di generare dati utili per lo sviluppo di applicazioni innovative, fornendo informazioni utilizzabili per ulteriori analisi che mirano a sfruttare la conoscenza presente sul web per creare nuove opportunità di ricerca e pianificazione. La flessibilità di questo

sistema lo rende particolarmente adatto ad essere implementato anche al di fuori del contesto milanese, potenzialmente in altre città, sia a livello nazionale che internazionale.

Come discusso in precedenza, il sistema può essere facilmente ampliato attraverso l'integrazione di ulteriori fonti di dati, come le piattaforme di social media; questi canali offrono una vasta quantità di informazioni sugli eventi che potrebbero non essere presenti sui tradizionali siti web. Inoltre, il sistema può essere adattato per elaborare formati di dati diversi, come immagini o contenuti multimediali, arricchendo ulteriormente il database e migliorando la qualità delle informazioni ottenute. In conclusione, il progetto costituisce una base solida per attività future, offrendo ampie possibilità di espansione e applicazioni in diversi settori. Questa prospettiva non solo conferma l'efficacia del sistema, ma apre anche la strada a nuovi sviluppi e miglioramenti.

Bibliografia

[1] Aavache (2023). Llmwebcrawler: A web crawler based on large language models. https://github.com/Aavache/LLMWebCrawler.

[2] OpenAI (2024). Openai models documentation. https://platform.openai.com/docs/models/.

[3] Unclecode (2023). Crawl4ai: Web crawling for ai applications. https://github.com/unclecode/crawl4ai.