UNIVERSITÀ
DI PARMA

Dipartimento Di Ingegneria e Architettura

Corso di Laurea Triennale in Ingegneria delle Tecnologie Informatiche

Confronto di modelli LLM e Transformer per la classificazione di articoli di giornale riguardanti incidenti stradali.

Comparison of LLM and Transformer models for the classification of newspaper articles regarding road accidents.

Relatore: prof. Michele Tomaiuolo

Federico Filice

Mat. 344387

Anno Accademico 2024-2025

A Meggie e nonna Assunta

Indice

1. Introduzione

La sicurezza stradale costituisce oggi un tema di forte rilevanza pubblica, poiché incide direttamente sulla qualità della vita dei cittadini e sul benessere delle comunità urbane. Infatti, gli incidenti stradali causano un impatto socioeconomico significativo. Secondo i dati ISTAT: nel 2024 si sono verificati 173.364 incidenti, che hanno causato 3.030 vittime e 223.853 feriti. Il report ACI-ISTAT stima, inoltre, che il costo sociale complessivo legato all'incidentalità stradale abbia raggiunto i 22,3 miliardi di euro nel 2023, impattando negativamente sulla vita di molte famiglie e indirettamente sul sistema economico. [1] [2]

Un elemento particolarmente critico, riguarda gli utenti vulnerabili della strada. Il 50% delle vittime appartiene infatti alle categorie fragili, come: pedoni, ciclisti e motociclisti. Tra le principali cause degli incidenti emergono: distrazione, velocità eccessiva e mancato rispetto delle precedenze, che insieme rappresentano il 37,8% dei casi.

In parallelo, le città stanno affrontando sfide crescenti legate all'aumento del traffico, all'inquinamento e alla necessità di sviluppare una mobilità più sostenibile. Oltre al potenziamento del trasporto pubblico al fine di ridurre il numero di veicoli privati su strada, un ruolo fondamentale è ricoperto dalla disponibilità di informazioni chiare, aggiornate e dettagliate sulla sicurezza stradale. La loro rappresentazione, in forma intuitiva e accessibile, può supportare tecnici e amministrazioni pubbliche nell'attuazione di decisioni consapevoli e interventi mirati.

Il lavoro di tesi, qui presentato, si inserisce all'interno di un percorso di ricerca sviluppato negli ultimi anni presso l'Università di Parma, ma trae ispirazione anche da progetti e ricerche di livello nazionale e internazionale. Tra questi si possono citare *Atlante degli incidenti ciclistici* realizzato dal Politecnico di Milano e le svariate mappe interattive elaborate da *Bike Ottawa*. Sono esempi di come dati, analizzati e adeguatamente rappresentati, possano mostrare in modo immediato aspetti della pericolosità stradale e diventare strumenti utili nella vita quotidiana.

In questo contesto, diventa importante affiancare ai dati ufficiali — talvolta obsoleti o non facilmente reperibili — informazioni complementari in grado di restituire un quadro più completo del territorio. Tra le fonti potenzialmente utili rientrano gli articoli di cronaca locale, che offrono un flusso continuo di notizie sugli incidenti, pur essendo privi di una struttura che ne consenta l'analisi sistematica.

Abstract:

Negli ultimi anni, l'avanzamento tecnologico nel campo dell'*intelligenza artificiale* ha creato nuove possibilità per analizzare automaticamente grandi quantità di testo non strutturato. Il lavoro qui presentato studia l'applicazione di *Large Language Model* (LLM) per l'analisi di articoli giornalistici riguardanti incidenti stradali. valutando l'efficacia di questi modelli sia come strumento di classificazione binaria, sia come

componente metodologica per il calcolo automatizzato dell'indice di pericolosità su un territorio definito. L'interesse verso tecniche di *Natural Language Processing* (NLP) deriva dalla loro utilità nell'effettuare *data mining*, ovvero estrarre conoscenza latente da grandi insiemi di dati, particolarmente utile quando le fonti ufficiali risultano incomplete o poco accessibili.

Da questa considerazione deriva il primo obiettivo della tesi: verificare se un *LLM* di dimensioni contenute e non specializzato sul task, nello specifico *Qwen3 – 4B Instruct*, possa costituire un'alternativa valida a modelli fine-tuned come *Italian BERT XXL* per la classificazione frase per frase tramite cinque etichette binarie (localizzazione, tempo, gravità, presenza di motocicli e di utenti deboli).

Il secondo obiettivo del lavoro riguarda l'impiego dei dati estratti tramite *LLM* per la creazione di una mappa territoriale raffigurante l'indice di pericolosità stradale del comune di Parma. In particolare, si intende valutare se le informazioni provenienti dalla testata giornalistica possano essere geolocalizzate e confrontate con il dataset ufficiale redatto dalla polizia locale, con lo scopo di verificare quanto la rappresentazione ottenuta da fonti non ufficiali si avvicini a quella fornita dagli *OpenData*.

La sperimentazione è stata articolata in tre fasi. La prima ha riguardato la progettazione e selezione dei prompt per migliorare le performance del modello. Una fase di preselezione, basata su un dataset ridotto, ha portato alla creazione di oltre 40 prompt tramite tecniche di *prompt engineering*. Ogni prompt è stato eseguito tre volte, così da ottenere la media delle metriche e stimare la deviazione standard per valutarne stabilità e robustezza. I migliori sono stati poi testati nell'ambiente finale per identificare quello da utilizzare nelle fasi successive.

La seconda fase della sperimentazione ha confrontato le prestazioni di *Qwen3 – 4B Instruct* con quelle di *Italian BERT XXL*. Lo scopo era di osservare se un *LLM,* di dimensioni contenute e non addestrato sul task, potesse avvicinarsi o superare le prestazioni di un modello più semplice, basato su *Transformer* e sottoposto ad un processo di fine-tuning su più di 130000 frasi. L'analisi ha evidenziato come *BERT* mantenga un vantaggio significativo, seppur relativamente ridotto, nella maggior parte delle etichette. In generale, emerge che un modello dedicato e sottoposto a *fine-tuning*, come *Italian BERT XXL*, mantiene un vantaggio significativo rispetto al modello *Qwen3 – 4B Instruct*. Questo suggerisce che l'impiego di un *LLM* di dimensioni maggiori potrebbe risultare in performance migliori, senza richiedere un processo di addestramento dedicato.

La terza fase ha descritto il processo di costruzione della pipeline adibita alla mappatura dell'indice di rischio stradale. Pipeline in cui, attraverso scraping del giornale online *ParmaToday*, sono stati raccolti 1777 articoli, suddivisi in 10868 frasi ed etichettati tramite *Qwen*, eseguito grazie all'utilizzo dell'infrastruttura di High Performance Computing (HPC) di ateneo. Successivamente, gli incidenti sono stati geolocalizzati tramite *NER* per estrarre le entità di localizzazione e *Nominatim* così da ottenere le coordinate. Infine, è stata creata una griglia geografica con l'indice di pericolosità stradale calcolato per entrambi i dataset, consentendo un confronto punto per punto. Il

confronto ha evidenziato una correlazione moderata (0,4) e un accordo discreto secondo l'analisi di *Bland-Altman*, risultati coerenti con la natura asimmetrica di record tra il dataset del comune di Parma e quello ricavato dalla stampa. Nel complesso, i risultati indicano che, con ulteriori migliorie e nuovi metodi discussi successivamente, questo approccio potrebbe costituire uno strumento informativo utile per il tema della sicurezza stradale.

In conclusione, il lavoro dimostra che l'unione tra tecniche *NLP*, geolocalizzazione e analisi statistica può trasformare articoli di cronaca non strutturati in dati organizzati, permettendo l'estrazione di conoscenza latente. Analisi future potrebbero risultare utili al fine di monitorare la sicurezza stradale e pianificare interventi mirati ed efficienti.

2. Background e strumenti

2.1. Stato dell'arte

2.1.1. Incidenti stradali, OpenData del Comune di Parma

Il progetto *OpenData* del Comune di Parma, condotto dalla Polizia Locale, ha reso disponibile pubblicamente un dataset contenente metadati relativi agli incidenti stradali verificatisi nel comune di Parma e avvenuti nel periodo tra il 2016 e il 2022. [3]

Il sito web che ospita il dataset offre la possibilità di visualizzare gli incidenti tramite una mappa cartografica, come illustrato nella Figura 2.1. Per ciascuna istanza è possibile visualizzare informazioni dettagliate come: data, ora, latitudine, longitudine, condizioni meteorologiche, presenza di illesi, feriti, morti, numero di persone coinvolte e molti altri parametri descrittivi dell'incidente.

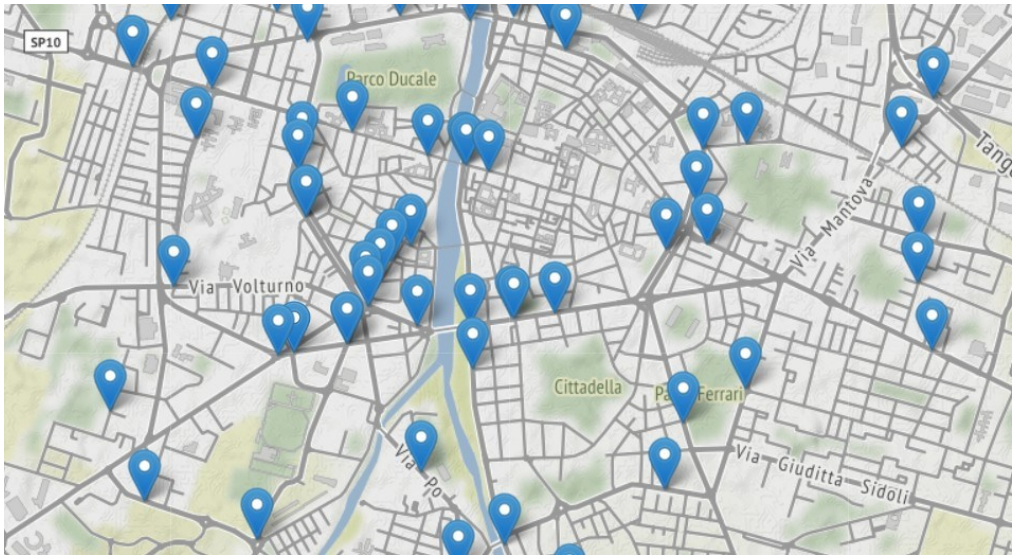

Figura 2.1: Mappa del progetto Incidenti stradali 2022 del Comune di Parma

Il dataset, popolato da 5162 incidenti, è stato utilizzato come benchmark per questa sperimentazione, considerando i dati al suo interno esaustivi e affidabili.

2.1.2. Ricerche precedenti

Diverse ricerche pregresse hanno contribuito a delineare il progetto di questo lavoro.

Il progetto di tesi di Elena Azzi "*DangeRoads*" ha come obiettivo la creazione di una mappa di geolocalizzazione degli incidenti stradali nel comune di Parma. Questo viene ottenuto tramite attività di *scraping* sul sito web *ParmaToday* e mediante l'impiego di

BERT per la classificazione delle frasi degli articoli di giornale. Il modello di *deep learning* è stato sottoposto a *fine-tuning* utilizzando un insieme di 900 frasi tratte da articoli, precedentemente classificate manualmente. [4]

La tesi sperimentale di Brianti si articola in due obiettivi principali: lo sviluppo di un classificatore binario di frasi basato sul modello di *deep learning* linguistico *BERT* e la realizzazione di una mappa della pericolosità stradale del Comune di Parma. Quest'ultima è stata ottenuta mediante la manipolazione dei dati classificati, a partire, da un dataset di articoli di giornale estratti dalla collana *ParmaToday*. [5]

Il resoconto dell'internato di laboratorio svolto da Cavazzuti ha analizzato il comportamento del *LLM Llama3.2 3B Instruct* nell'etichettatura binaria di articoli di giornale riguardanti incidenti stradali ed estratti dalla testata giornalistica *ParmaToday*. La sperimentazione ha interessato un test di etichettatura basato sull'analisi frase per frase e un altro test utilizzando l'articolo intero. Il risultato di questa comparazione è che l'analisi per articolo rifletteva una miglior accuratezza rispetto all'altra metodologia. [6]

Le ricerche sopracitate sono state utili perché hanno consentito di usufruire del dataset etichettato manualmente da Azzi al fine di confrontare le performance rispetto alle tecnologie impiegate da Brianti, adottando la stessa metodologia, ma un modello differente. Per rendere il confronto il più veritiero possibile è stata usata la modalità di etichettatura frase per frase, nonostante il lavoro di Cavazzuti ne evidenziasse l'inferiorità rispetto alla modalità per articolo intero.

2.1.3. Atlante degli incidenti ciclistici

Il Politecnico di Milano ha recentemente pubblicato una ricerca contenente cinque dashboard interattivi e relativi agli incidenti ciclistici in tutta Italia. I dati sono stati forniti dall'ISTAT e riguardano incidenti tra il 2022 e il 2023. Quest'analisi è stata realizzata con l'obiettivo di informare e contribuire a miglioramenti futuri nella sicurezza ciclistica in Italia. [7]

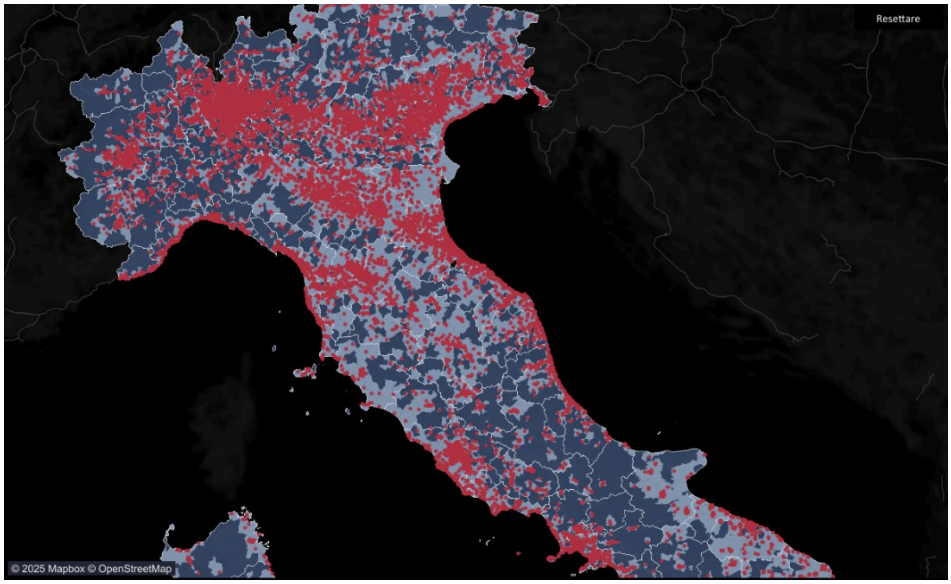

Figura 2.2: "Mappa degli incidenti ciclistici – Ciclisti feriti 2022 e 2023", Politecnico di Milano

2.1.4. *Cycling level of traffic stress*, *Collision map*

Bike Ottawa, gruppo canadese per la difesa dei diritti, gestisce un sito contenente diverse mappe interattive. Tra queste mappe, alcune sono particolarmente interessanti per la ricerca: "*Cycling Level of Traffic Stress*" che classifica in una mappa le strade secondo quattro livelli di stress per i ciclisti e "*Collision map*",una mappa degli incidenti stradali basati sugli open data forniti dalla città di Ottawa.

Successivamente, tutte le mappe prodotte dal progetto sono state integrate in un *Route Planner*, che permette agli utenti di pianificare percorsi ciclistici personalizzati. [8]

Figura 2.3: "Collision map, All modes" anno 2019, Bike Ottawa

I progetti del Politecnico di Milano e di Bike Ottawa sono stati di ispirazione per la realizzazione di questo lavoro.

2.2. Strumenti e tecnologie utilizzate

2.2.1. BERT

BERT, acronimo di *Bidirectional Encoder Representations from Transformers*, è un modello di *deep learning* sviluppato da *Google AI Language* per il preaddestramento di sistemi di *NLP*. Si basa su un'architettura *Transformer bidirezionale*, che permette al modello di comprendere il contesto di una parola sia dalla parte a sinistra che da quella a destra all'interno di una frase. [9]

In particolare, nella ricerca precedente è stato impiegato *Italian BERT XXL cased*, un modello preaddestrato, che è stato sottoposto a *fine-tuning* per 5 epoche di addestramento su 132.946 frasi etichettate automaticamente. [10]

In questo elaborato, i risultati ottenuti utilizzando *BERT* nei lavori precedenti sono stati confrontati con quelli ottenuti utilizzando il *LLM Qwen*, con la stessa metodologia, al fine di valutare le differenze di prestazione tra i due approcci.

2.2.2. Qwen

Qwen è la famiglia di *LLM* sviluppata da *Alibaba Cloud*. In particolare, sono stati utilizzati modelli *open-weights*, ossia modelli i cui pesi addestrati sono pubblicamente disponibili ed eseguibili localmente. Sebbene rilasciati con licenza *open-source* (*Apache 2.0*), non risultano pienamente trasparenti poichè il codice di addestramento e i dataset utilizzati per il training non sono pubblicati né documentati in dettaglio. [11]

In questa sperimentazione sono state utilizzate due versioni diverse di *Qwen* ottenute dal sito *Hugging Face*:

- *Qwen2.5 – 1.5B Instruct*, costituito da 1.54 miliardi di parametri. [12]
- *Qwen3 – 4B Instruct*, composto da 4 miliardi di parametri. [13]

Le differenze principali tra i due modelli riguardano la dimensione, perché il modello con il maggior numero di parametri possiede maggiori capacità di comprensione, migliori performance e anche la generazione, poiché la generazione 3 ha maggiori capacità rispetto alla 2.5. [14]

Entrambi i modelli sono stati utilizzati senza effettuare fine-tuning. *Qwen2.5 – 1.5B Instruct* è stato adottato durante la valutazione comparativa preliminare, nella quale sono stati testati oltre 40 prompt per valutarne le prestazioni generali. Questo *LLM* è stato preferito al fine di velocizzare le analisi, nonostante sia meno complesso e preciso rispetto a quello utilizzato successivamente. Invece, *Qwen3 – 4B Instruct* è stato impiegato per la selezione finale degli otto prompt con le migliori performance e per tutte le analisi successive.

Non sono stati considerati modelli di dimensioni maggiori per mantenere un confronto equo con *BERT*, che presenta una complessità computazionale più contenuta.

2.2.3. High Performance Computing

Per l'esecuzione degli script di etichettatura basati su *Qwen3 – 4B Instruct*, è stata utilizzata l'infrastruttura di *HPC* (*High Performance Computing*) dell'Università degli Studi di Parma e messa a disposizione dal Servizio di Calcolo Scientifico per attività di ricerca e sperimentazione computazionale. [15]

Il sistema *HPC* è basato su un'architettura cluster con nodi CPU e GPU, gestiti tramite il job scheduler *SLURM*, che consente un'allocazione efficiente delle risorse e una gestione ottimizzata dei carichi di lavoro.

L'impiego dell'*HPC* di Ateneo ha reso possibile eseguire lo script di etichettatura tramite il modello *Qwen3 - 4B Instruct*, garantendo risorse adeguate alla gestione dei carichi computazionali elevati e assicurando tempi di elaborazione ridotti.

2.2.4. NER SpaCy

La *NER*, acronimo di *Named Entity Recognition*, è una tecnica di estrazione delle informazioni che consente di individuare e classificare entità specifiche all'interno di un testo non strutturato, cioè, scritto in linguaggio naturale. [16]

Per questa sperimentazione è stato utilizzato *it_nerIta_trf* della libreria *SpaCy*. Questo modello, sviluppato da *BullMount*, è basato su un'architettura *Transformer* che sfrutta *BERT* ed è preaddestrato sui testi in lingua italiana. [17]

L'impiego di questo modello è stato necessario per riconoscere ed estrarre le componenti di localizzazione nei testi degli articoli di giornale.

2.2.5. Nominatim

OpenStreetMap è un progetto collaborativo basato sul contributo volontario e con lo scopo di creare una cartina geografica globale liberamente accessibile. Il progetto offre diversi strumenti e dataset, tutti sotto licenza libera, consentendone l'utilizzo gratuito. [18]

In questo lavoro è stata utilizzata *l'API* (Application Programming Interface) di geolocalizzazione *Nominatim*, che consente di ottenere le coordinate geografiche di un luogo a partire da nomi di località o indirizzi espressi in forma testuale. [19]

Questo servizio è stato utilizzato per ricavare le coordinate geografiche associate alle entità di localizzazione identificate tramite il modello di *NER*.

2.2.6. Librerie utilizzate

Durante questo progetto sono state utilizzate diverse librerie in ambiente virtuale Python, sotto sono elencate le più importanti:

Scikit-learn: libreria di *machine learning open-source* che offre diversi algoritmi e strumenti come la pre-elaborazione dei dati, la selezione e la valutazione dei modelli. In questo lavoro è stata utilizzata per il calcolo di metriche come accuratezza, *F1-score medio* e *F1-score ponderato*. [20]

Python requests: libreria Python creata per inviare richieste *HTTP* in modo semplice, senza dover aggiungere manualmente query o parametri agli *URL*. Nel progetto è stata utilizzata per lo *scraping* degli articoli di giornale di *ParmaToday*. [21]

Beautiful Soup 4: libreria Python per il *parsing* di documenti *HTML* e *XML*, che consente di trasformare il documento in una struttura ad albero navigabile e di estrarre facilmente elementi specifici. Nel progetto è stata utilizzata per estrarre *URL* e tutte le informazioni utili relative agli articoli di giornale ottenuti tramite *scraping*. [22]

Geopy: fornisce strumenti per la geocodifica, consentendo di ottenere coordinate geografiche a partire da indirizzi, città, luoghi e viceversa. Nella sperimentazione è stata impiegata per ricavare un punto situato a una distanza predefinita da un altro punto, entrambi espressi tramite latitudine e longitudine. [23]

Shapely: libreria per la manipolazione e l'analisi di oggetti geometrici piani e basata su altre librerie *open-source. Shapely* è stata utilizzata per creare il confine del comune di Parma tramite coordinate, permettendo di distinguere gli incidenti avvenuti all'interno del comune da quelli avvenuti al suo esterno. [24]

Folium: libreria per la visualizzazione di dati su mappe interattive basate su *OpenStreetMap* e *JavaScript*. Consente la creazione di mappe, layers e l'esportazione dell'output in formato *HTML*. In questo progetto è stato utilizzato per generare mappe interattive al fine di rappresentare in modo intuitivo l'indice di pericolosità stradale. [25]

Matplotlib: libreria per la visualizzazione dei dati che permette la creazione di grafici statici, con la possibilità di renderli interattivi. Questa libreria è utilizzata per creare tutti i grafici necessari alle analisi quantitative e comparative. [26]

3. Architettura

Questo lavoro sperimentale si è posto l'obiettivo di utilizzare il *LLM Qwen3 – 4B* per etichettare, secondo parametri definiti in seguito, articoli riguardanti incidenti stradali analizzati frase per frase.

Nella fase iniziale è stato necessario selezionare i prompt più efficaci mediante tecniche di *prompt engineering* per ottenere risultati migliori in quanto non è stato previsto un processo di *fine-tuning*.

Successivamente, i risultati dell'etichettatura sono stati confrontati con quelli ottenuti tramite il modello *BERT*, precedentemente sottoposto a *fine-tuning* per il task specifico.

Infine, è stato eseguito il confronto tra il dataset sperimentale e i dati di riferimento di *OpenData* del Comune di Parma tramite analisi statistiche e visualizzazioni.

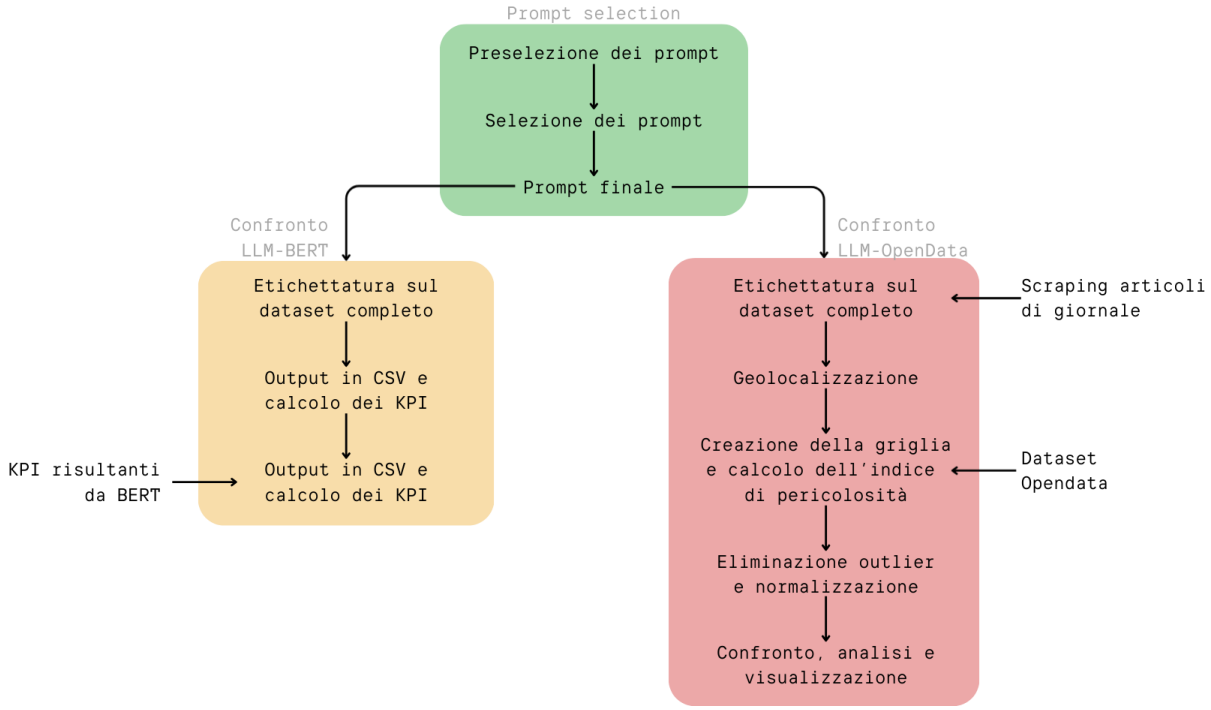

Figura 3.1: Mappa concettuale raffigurante le parti principali della sperimentazione

3.1. Prompt selection

3.1.1. Obiettivo

L'obiettivo della fase di *prompt selection* è quello di individuare, tra diversi candidati, il prompt con performance migliori.

In primo luogo, è stato necessario creare e raffinare i prompt da sottoporre a confronto. La valutazione è stata eseguita applicando tecniche e principi di *prompt engineering* come: chiarezza, precisione, definizione del contesto, struttura dell'output, vincoli, ma anche tipologie di prompting, tra cui role playing, *zero-shot* e *few-shot*. [27]

Successivamente, è stata valutata la possibilità di effettuare il confronto ripetendo ciascun esperimento associato a un prompt per tre volte, in modo da calcolare una stima della deviazione standard e dei *Key Performance Indicators* (KPI) associati ad ogni prompt.

Infine, la valutazione dei prompt è stata eseguita tramite i *KPI* descritti successivamente.

3.1.2. Prompt

L'obiettivo del prompt di etichettatura è la corretta classificazione delle seguenti cinque classi binarie: Localizzazione, Tempo, Grave, Moto e Utenti deboli.

Il modello riceve una frase non etichettata in input e ha il compito di impostare a 1 le etichette trovate al suo interno, lasciando a 0 quelle non individuate. Infine, deve fornire l'output formato dai 5 parametri etichettati nel formato *JSON*.

Un esempio è inserito successivamente:

```
Input:
"Durante la mattinata di ieri un'automobile ha investito un
signore mentre attraversava la strada, fortunatamente
l'anziano non ha riscontrato ferite gravi."
Output:
{
"localizzazione": 0,
"tempo": 1,
"grave": 0,
"moto": 0,
"utenti_deboli": 1
}
```

Key Performance Indicators

Al fine di valutare le performance di un prompt, sono state calcolate le seguenti metriche per ogni etichetta:

- *F1-score medio*: Calcola l'F1-score per ciascuna etichetta e poi calcola la media aritmetica semplice. Non tiene conto di sbilanciamenti di etichette. [28]

$$F1_{macro} = \frac{1}{k}\sum_{i=1}^{k} F1_i$$

$k =$ Numero di etichette

$F1_i =$ F1-score dell'etichetta i

- *F1-score ponderato*: Calcola l'F1-score per ciascuna etichetta e applica una media ponderata pesata per il supporto, cioè il numero di istanze per ogni etichetta. Tiene conto di squilibri tra etichette. [28]

$$F1_{weighted} = \sum_{i=1}^{k} \omega_i \cdot F1_i, \quad \omega_i = \frac{n_i}{N}$$

$n_i =$ Numero campioni della classe i

$N =$ Numero totale di campioni

- *Deviazione standard*: misura il grado di dispersione dei valori rispetto alla media. [29]

$$\sigma_X = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_X)^2}$$

$x_i =$ Valori individuali

$\mu =$ Media della popolazione

- *Coefficiente di variazione percentuale*: è una misura della variabilità relativa di un insieme di dati. È stato calcolato per *F1-score medio* e per *F1-score ponderato*. [30]

$$CV\% = \frac{\sigma_X}{F1} \cdot 100$$

Per garantire obiettività, ripetibilità e calcolare la metrica minima di comparazione tra prompt, ogni esperimento è stato ripetuto tre volte, in modo da ottenere una media dei *KPI* e una stima della deviazione standard.

Modalità e tipologie utilizzate

Le due modalità di etichettatura utilizzate sono:

- *Multilabeling*: modalità in cui un singolo prompt richiede l'etichettatura di tutti i parametri.
- *Monolabeling*: modalità in cui viene fornito un singolo prompt per ogni etichetta separatamente, rendendo indipendente l'analisi delle singole etichette.

Le metodologie di prompting adottate durante la sperimentazione sono: [27]

- *Zero-shot*: tecnica di prompting in cui al modello viene assegnato un compito senza fornire alcun esempio esplicito su come svolgerlo.
- *Few-shot*: tecnica di prompting in cui, oltre al lavoro assegnato, vengono forniti uno o più esempi del compito da svolgere, così da guidarne il comportamento ed il formato di output.

La metodologia *few-shot* è stata testata con 2, 4, 6, 8 esempi, organizzati a coppie formate da esempio negativo e positivo.

Nello specifico, sono state testate le quattro combinazioni derivanti dall'incrocio tra modalità di etichettatura e tipologie di prompting:

- *Multilabel zero-shot*
- *Monolabel zero-shot*
- *Multilabel few-shot*
- *Monolabel few-shot*

3.1.3. Preselezione dei prompt

La fase di preselezione dei prompt è stata necessaria poiché i LLM risultano particolarmente sensibili alla formulazione del prompt, alla struttura delle domande e alla scelta di specifiche parole.

Sono stati testati complessivamente 43 prompt di etichettatura, con un minimo di 10 prompt per ciascuna tipologia. A causa dell'elevato numero di iterazioni richieste dagli esperimenti, è stato utilizzato il modello *Qwen2.5 – 1.5B Instruct*, eseguito sulla piattaforma *Google Colab* all'interno di un ambiente virtuale Python. La scelta di questo modello è motivata dal numero minore di parametri rispetto a quello impiegato nella fase di confronto finale.

Il dataset utilizzato per la preselezione, preparato in precedenza da Elena Azzi, è composto da 900 frasi provenienti da articoli di giornale ed etichettate manualmente. Al fine di velocizzare questo processo è stato deciso di utilizzare un sottoinsieme del dataset, formato da 200 frasi.

Queste scelte hanno permesso di ridurre i tempi e i costi computazionali della fase preliminare, rendendo possibile il confronto di prestazioni per ogni prompt e la scelta dei più performanti.

3.1.4. Selezione del prompt finale

Le nuove condizioni di testing sono state: il dataset completo di 900 frasi e il modello *Qwen3 – 4B Instruct*, eseguito localmente tramite l'infrastruttura di HPC di ateneo in un ambiente virtuale in linguaggio Python.

Al termine della fase di preselezione, e seguendo lo stesso procedimento descritto in precedenza, sono stati selezionati gli otto prompt migliori per una seconda fase di sperimentazione.

I prompt testati sono i seguenti:

- Un prompt *multilabel zero-shot*.
- Un prompt *monolabel zero-shot*.
- Un totale di quattro prompt *monolabel few-shot*, rispettivamente a 2, 4, 6, 8 esempi, per analizzare l'impatto del numero di esempi sulle performance.
- Due prompt *multilabel few-shot*, con 4 e 8 esempi.

3.2. Confronto con BERT

Il confronto tra le performance del modello *Qwen3 - 4 B Instruct* e del modello *Italian BERT XXL* è stato eseguito sul dataset da 900 frasi, già utilizzato per la selezione del prompt migliore. L'obiettivo della valutazione è di determinare se un modello più semplice e leggero, come *BERT* — addestrato per il task specifico tramite *fine-tuning* — possa risultare più o meno performante rispetto a un *LLM general-purpose* — non sottoposto a *fine-tuning* — che possiede una dimensione di 4 miliardi di parametri, relativamente contenuta se confrontato con i modelli di grandi dimensioni attualmente disponibili sul mercato, che raggiungono centinaia di miliardi di parametri.

BERT, il modello introdotto nel 2018 da *Devlin et al.*, è stato sviluppato da *dbmdz* e pre-addestrato su un dataset di 81 Gb contenente l'interezza di wikipedia in italiano e il corpus *OSCAR*. Nello specifico è stato utilizzato *Italian BERT XXL Cased*, un modello formato da 111 milioni di parametri, che riconosce la differenza tra lettere maiuscole e minuscole ed è stato sottoposto a fine-tuning per 5 epoche su 132.946 frasi etichettate automaticamente. [5] [31]

BERT è un modello di deep learning *encoder-only*, progettato per compiti predittivi come la classificazione o l'estrazione di informazioni. Si basa sull'architettura *Transformer*, costituita da un insieme di moduli che operano in sequenza: [9]

1. *Tokenizer*: converte il testo in *token*, ossia in sequenze di interi che rappresentano le unità linguistiche di input.
2. *Embedding*: trasforma i *token* in vettori di valori reali collocati in uno spazio vettoriale a n dimensioni. In questo spazio, *token* con significati simili tendono a presentare vettori vicini.
3. *Encoder*: è il cuore del modello ed è responsabile dell'analisi bidirezionale del contesto. Grazie ai meccanismi di *self-attention*, l'encoder identifica relazioni semantiche e sintattiche tra i token ed estrae rappresentazioni profonde del testo.
4. *Task head*: modulo aggiunto durante il *fine-tuning* che utilizza le rappresentazioni prodotte dall'encoder per svolgere il compito richiesto. Non genera testo, ma produce output strutturati.

Mentre *Qwen3*, un modello *GPT* (Generative Pre-trained Transformer) di tipo *decoder-only*, è stato introdotto nel 2025. Nello specifico è stata utilizzata la versione a 4 miliardi di parametri senza *fine-tuning* e in configurazione *non-thinking*. [11]

Come *BERT*, *Qwen* è basato su un'architettura *Transformer* composta da un *tokenizer* e da uno strato di *embedding*, ma differisce nelle componenti successive: [14]

- *Decoder*: elabora il testo tramite meccanismo di *self-attention unidirezionale*, che calcola il peso di un *token* rispetto a tutti e soli i *token* precedenti nella sequenza. A differenza dell'encoder bidirezionale di *BERT*, il modello considera esclusivamente il contesto a sinistra del *token* in elaborazione. In questo modo è

in grado di prevedere i *token* più probabili che potrebbero comparire nella sequenza.

- *Task head generativa*: produce le previsioni in modalità token per token. In questa sperimentazione è stata impiegata per un compito di classificazione, interpretando l'output generativo, richiesto in formato strutturato, come etichettatura binaria.

2.3. Confronto con OpenData

3.2.1. Obiettivo

L'obiettivo del confronto tra il dataset sperimentale, ottenuto dagli articoli di *ParmaToday*, e il dataset di benchmark, fornito dagli *OpenData* del Comune di Parma, è quello di valutare il livello di accordo e la correlazione tra i due metodi di misura. Tale analisi permette di verificare se l'elaborazione automatica degli articoli di giornale proposta in questo lavoro, possa produrre una mappatura realistica dell'indice di pericolosità stradale, quindi comparabile con quella derivante dai dati ufficiali.

Prima di eseguire un confronto sistematico è stato necessario sviluppare una pipeline di preparazione per i dataset. Di seguito è riportata una descrizione delle fasi principali:

- Pre-elaborazione, applicata al solo dataset sperimentale.
- Preparazione al confronto, ha interessato entrambi i dataset.
- Confronto statistico e visualizzazione.

3.2.2. Pre-elaborazione

La necessità di iniziare da una fase di *preprocessing* deriva dalla natura del dataset sperimentale: i dati in input sono formati da un testo non strutturato e altre informazioni non rilevanti per l'analisi. Il dataset di output desiderato al termine del *preprocessing* è quindi formato esclusivamente dalle componenti necessarie alla verifica dei dati (ID, testo, posizione) e alle modifiche successive (latitudine, longitudine, grave).

Preparazione del dataset sperimentale

Il dataset iniziale è un file di testo contenente 1777 articoli, ciascuno composto da: titolo, data di pubblicazione, URL e corpo dell'articolo.

Tale dataset è stato ottenuto tramite un apposito programma di *scraping*, creato per raccogliere automaticamente gli articoli riguardanti incidenti stradali dalla sezione cronaca di *ParmaToday* nel periodo 2016-2022, esattamente il periodo di riferimento del dataset *OpenData*.

Infine, gli articoli del dataset sperimentale sono stati divisi frase per frase per consentire un'etichettatura binaria coerente con la metodologia applicata tramite *BERT*.

Etichettatura e scrematura dei risultati

Dopo la preparazione, il dataset è stato processato tramite lo script di etichettatura definito nella fase di prompt selection, utilizzando *Qwen3 – 4B Instruct*.

Dato che le analisi successive richiedevano solo alcune componenti del dataset, è stata eseguita una scrematura. In questo modo il numero di righe del file è diminuito da 10868 a 1777 e le analisi successive sono risultate molto più efficienti.

Geolocalizzazione

La geolocalizzazione aveva l'obiettivo di ottenere le coordinate dei luoghi degli incidenti a partire dalle frasi in cui l'etichettatura binaria precedente ha individuato informazioni di localizzazione. Il processo si è articolato in tre fasi principali:

Nella prima fase è stata utilizzata la *NER* per estrarre automaticamente le entità di localizzazione nella frase scelta, ad esempio "via Parigi", "Stazione" o "strada Argini".

Successivamente, sono state ricavate le coordinate di latitudine dalle entità appena ottenute tramite l'*API Nominatim*.

Nella terza fase, i risultati sono stati filtrati nuovamente per includere esclusivamente gli incidenti avvenuti nel comune di Parma, per mantenere coerenza con il dataset di benchmark, che contiene solo informazioni relative al comune.

3.2.3. Preparazione al confronto

Da questo punto in avanti, tutte le operazioni sono state eseguite in parallelo su entrambi i dataset.

Per ottenere risultati confrontabili punto per punto, è stata generata una griglia di punti a distanza costante all'interno del territorio comunale. Questa scelta è stata necessaria in quanto i due dataset sono sbilanciati per numerosità degli incidenti riportati e gli articoli giornalistici non forniscono la posizione esatta dell'incidente, rendendo impossibile il confronto diretto tra le coordinate appena ottenute.

Per ciascun punto della griglia è stata decisa un'area entro la quale ricercare gli incidenti da attribuire a quel punto. Così è stato possibile calcolare l'indice di pericolosità stradale dell'area prestabilita e ottenere un confronto.

3.2.4. Confronto statistico e visualizzazione

Prima di procedere con le analisi statistiche, è stata verificata la tipologia di distribuzione dei dati e, sulla base dei risultati ottenuti, sono stati gestiti gli eventuali outlier in modo appropriato.

Per semplificare l'analisi è stato creato un dataset unico contenente, per ciascun punto della griglia, i due valori dell'indice di pericolosità stradale calcolati rispettivamente dal dataset sperimentale e da quello di benchmark.

I grafici atti alla visualizzazione dei risultati dell'analisi statistica sono:

- *Scatter plot*: utilizzato per studiare l'eventuale relazione lineare tra i due metodi di misura. Inoltre, è stato calcolato l'indice di *correlazione lineare di Pearson* e il livello di significatività statistica, *p-value*.

- *Bland-Altman plot*: impiegato per valutare il livello di accordo tra due metodi di misura. Per questa visualizzazione è stato calcolato il *bias*, ovvero la differenza media di misurazione tra i due metodi.
- *Heatmap*: utilizzate per una valutazione intuitiva della distribuzione spaziale dei dati. Sono state create le seguenti *heatmap* tramite *folium*: Mappa degli incidenti nella provincia di Parma, e Mappa raffigurante l'indice di pericolosità stradale in griglia nel comune di Parma, entrambe formate dai dati provenienti dal dataset sperimentale, Mappa dell'indice di pericolosità stradale in griglia proveniente dagli *Opendata*, e infine, Mappa della sovrastima e sottostima tra i due metodi di analisi.

4. Implementazione

4.1. Selezione dei prompt

Successivamente verrà descritta l'implementazione e il funzionamento della fase di *prompt selection*.

4.1.1. Preselezione dei prompt

Come anticipato, nella fase di preselezione sono state testate quattro tipologie di etichettatura diverse: *multilabel zero-shot*, *monolabel zero-shot*, *monolabel few-shot* e *multilabel few-shot*. L'implementazione tecnica e i risultati associati a ciascuna modalità saranno discusse successivamente.

Per ogni prompt sono state effettuate tre esecuzioni indipendenti, in modo da stimare la deviazione standard delle performance. Questa scelta è motivata dall'osservazione che i risultati presentano un certo grado di variabilità, attribuibile alla natura probabilistica del modello, all'ambiguità intrinseca del linguaggio naturale e ad altri fattori non completamente osservabili. La ripetizione degli esperimenti, oltre a quantificare una stima della variabilità dei risultati tramite il calcolo della deviazione standard, ha permesso di calcolare la media dei KPI, fornendo una metrica di confronto più affidabile ed oggettiva tra i diversi prompt.

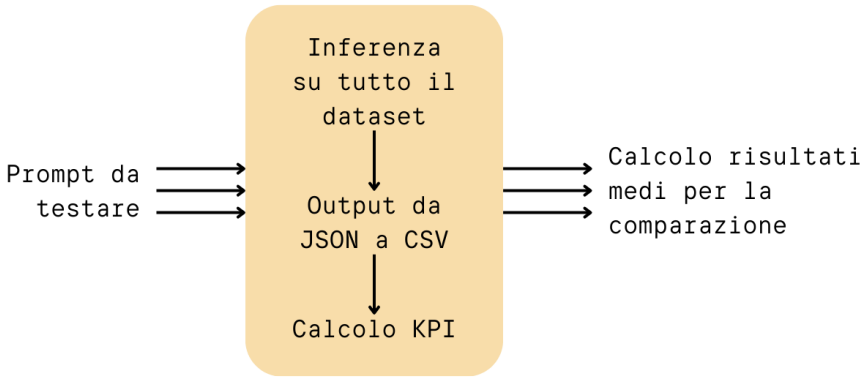

Figura 4.1: processo sul singolo prompt.

Il *LLM* utilizzato per l'etichettatura binaria, *Qwen2.5 – 1.5B Instruct*, è stato eseguito localmente sulla piattaforma *Google Colab* su un dataset di 200 frasi etichettate precedentemente al fine di calcolare i *KPI* necessari alla preselezione.

In questo script vengono impiegate alcune librerie fondamentali per l'elaborazione e la classificazione automatica del testo tramite *LLM*. In particolare, la libreria *transformers* di *Hugging Face* consente di utilizzare modelli di linguaggio per l'analisi semantica,

torch fornisce il supporto per il calcolo su GPU e la gestione dei tensori. Le librerie *csv*, *json*, *os* e *re* sono utilizzate per la gestione di file *CSV*, la manipolazione di dati in formato *JSON*, la gestione delle operazioni di sistema e la ricerca tramite espressioni regolari. Infine, viene utilizzata *scikit-learn* per calcolare i *KPI*, quali accuratezza e *F1-score.*

Algoritmo di selezione

In questa sezione verranno descritti i dettagli implementativi dell'algoritmo di selezione, con seguente sottolineatura delle differenze tra script monolabel e multilabel.

Per prima cosa, è necessario caricare il modello LLM nell'ambiente virtuale Python.

Il modello, pubblico e disponibile gratuitamente senza richiedere il *token* di accesso, è caricato dal sito *HuggingFace*. Nel caso in cui il LLM sia stato caricato dal sito, esso viene salvato localmente per un prossimo utilizzo. In questo modo, vengono ridotti i tempi di caricamento nelle esecuzioni successive alla prima. In *Figura 4.2*, viene presentato il frammento di codice interessato.

```
if os.path.exists(local_model_path):
  tokenizer = AutoTokenizer.from_pretrained(local_model_path)
  model = AutoModelForCausalLM.from_pretrained(
          local_model_path).to(device)
else:
  print("Scaricando modello per la prima volta")
  tokenizer = AutoTokenizer.from_pretrained(model_id)
  model = AutoModelForCausalLM.from_pretrained(
          model_id).to(device)
  tokenizer.save_pretrained(local_model_path)
  model.save_pretrained(local_model_path)
  print("Modello salvato")
```

Figura 4.2: Caricamento del modello LLM Qwen da HuggingFace.

L'input e output è nello stesso formato per tutti gli script utilizzati:

- Input: file in formato CSV contenente frasi da classificare, divisi in: frase; localizzazione; tempo; grave; moto; deboli.
- Output: dataset CSV contenente frasi classificate e divisione analoga all'input.

Di seguito è riportato lo pseudocodice che descrive i prompt multilabel.

```
1. Carica modello e tokenizer da HuggingFace:
   - Se il modello è presente in locale: caricalo
   - Altrimenti: scarica da HuggingFace e salva in locale

2. Per ogni frase nel dataset:
      a. Costruisci il prompt:
           - Inserisci istruzioni system e user
           - [Se few-shot: aggiungi N esempi annotati]
           - Inserisci la frase da etichettare

      b. Esegui l'inferenza:
           risposta ← LLM (prompt)

      c. Estrai il risultato:
           - Cerca un JSON nell'output
           - Normalizza e valida la struttura
           - Se valido: estrai valori binari
           - Altrimenti: assegna etichette a 0

      d. Output in CSV.
```

Figura 4.2: pseudocodice relativo al prompt multilabel, nelle versioni zero e few-shot.

Nel caso della tipologia monolabel, la struttura generale dell'algoritmo rimane invariata, ma la classificazione viene effettuata separatamente per ciascuna etichetta.

In particolare, per ogni frase del dataset, l'algoritmo entra in un ciclo interno che scorre tutte le etichette da assegnare: per ciascuna di esse viene costruito un prompt dedicato, e nel caso few-shot vengono aggiunti N esempi relativi alla specifica etichetta.

Il parsing dell'output differisce dalla modalità multilabel, poiché il modello genera un *JSON* contenente esclusivamente il valore binario della singola etichetta in esame. Il valore estratto viene temporaneamente memorizzato in un array tramite la funzione *append()*. Una volta elaborate tutte le etichette l'array risultante, che contiene i cinque valori in ordine, viene scritto come riga nel file *CSV* di output.

Il codice di *parsing* della risposta ottenuta dal *LLM* è riportato in *Figura 4.3*.

```
response = tokenizer.decode(outputs[0],
skip_special_tokens=True).strip()

result = {label: 'n'}

json_candidates = re.findall(r"\{.*?\}", response,
re.DOTALL)

for candidate in json_candidates:
  candidate_fixed = candidate.replace("'", '"')
    try:
      parsed = json.loads(candidate_fixed)
      if label not in parsed:
        continue
      if str(parsed[label]) not in ["0", "1"] and
                    parsed[label] not in [0, 1]:
        continue
      result = {label: int(parsed[label])}
      break
    except Exception:
      continue
return result[label] if result[label] in [0, 1] else 0
```

Figura 4.3: parsing dell'output di etichettatura di una frase in formato JSON.

Lavorando con i *LLM* è necessaria una gestione dell'output tramite *parsing* e validazione robusti, in quanto questi modelli non sono progettati per garantire output strutturati in formati specifici. In questo caso è stato utilizzato il formato *JSON*, poiché racchiude le informazioni tramite parentesi graffe, facilmente identificabili con una *ReGEx* (Regular Expression), ed è un formato rigido e quindi validabile con logica condizionale semplice. Di seguito è specificato il funzionamento del codice di *parsing*.

La *ReGEx* identifica tutti i possibili output *JSON* nella risposta del modello e li inserisce in una lista. Successivamente, tramite un ciclo for, vengono eseguiti i seguenti passaggi per ogni candidato:

- Vengono sostituiti eventuali apici singoli con doppi apici (errore riscontrato frequentemente durante la sperimentazione).
- Viene verificata la presenza della label richiesta nel *JSON* a cui è stato eseguito il *parsing*.
- Se il valore associato all'etichetta non è esclusivamente 0 o 1 (come stringa o numero intero), il formato non è valido e si passa al candidato successivo.
- Se tutte le validazioni hanno successo, il candidato viene salvato come risultato finale.

La gestione degli errori di *parsing* avviene tramite il lancio di eccezioni generiche, garantendo che *JSON* malformati non interrompano l'elaborazione. In caso nessun candidato risulti valido, viene restituito un valore di default (0) come strategia di *fallback*.

Multilabel zero-shot

Il prompt *multilabel zero-shot* rappresenta la configurazione più semplice tra le quattro modalità di analisi. In questa metodologia, viene fornito un singolo prompt senza fornire esempi, al fine di classificare tutte e cinque le etichette. In totale, sono state testate 10 varianti di questa tipologia, modificandone il testo.

Il prompt è stato strutturato in due parti distinte, in modo da controllare con maggior precisione il comportamento del modello: *System*, definisce le regole di alto livello, come il formato di output desiderato e i vincoli di risposta e *User* contiene l'istruzione vera e propria, ovvero il prompt di etichettatura. Questa struttura è stata mantenuta anche nelle modalità successive.

Nel corso della sperimentazione sono state iterate diverse versioni della parte di *System*, con l'obiettivo di esplicitare il formato di output richiesto e le regole di assegnazione delle etichette binarie affinché il *LLM* rispondesse nel formato corretto. In *Figura 4.4* è riportata la versione finale del prompt *System* impiegato:

```
{"role": "system",
"content": (
  "rispondi solo ed esclusivamente tramite un JSON valido."
  "usa come etichette: "
  "localizzazione, tempo, grave, moto, utenti_deboli"
  "i valori di questo json devono essere binari, quindi 0 o
1. non scrivere altro nella risposta."
)}
{"role": "user",
"content": (
  "Leggi questa frase di cronaca su un incidente stradale.
  rispondi alle domande che ti pongo con 1 se la risposta è
  si, 0 se la risposta è no:\n\n"
     "- Si capisce dove è successo?\n"
     "- Si capisce quando è successo?\n"
     "- L'incidente è grave?\n"
     "- C'entra una moto?\n"
     "- Ci sono persone vulnerabili coinvolte?\n\n"
  f"Frase: '{phrase}'\n"
  "Risposta:"
)}
```

Figura 4.4: Prompt 1 multilabel zero-shot, divisione in system e user.

Invece, la parte *User* del prompt rappresenta la componente sulla quale sono state effettuate diverse iterazioni, con l'obiettivo di migliorare le prestazioni del modello. In una prima fase sono stati sviluppati cinque prompt con approcci stilistici differenti:

1. Linguaggio informale con domande dirette in stile intervista.
2. *Role Playing*, in cui il modello assume il ruolo di un esperto del task ("sei un analista") e di conseguenza il testo è scritto in modo formale.

3. Approccio minimalista, basato su un formato sintetico simile a un dizionario, con istruzioni estremamente brevi e prive di contesto.
4. Stile misto, caratterizzato da un'alternanza casuale tra domande brevi e definizioni strutturate in forma di dizionario.
5. Analisi sequenziale, richiede di analizzare prima la frase e successivamente di assegnare le etichette, combinando domande dirette e indicazioni su quali elementi ricercare.

L'analisi dei primi risultati ha evidenziato che il p*rompt 1* forniva le performance migliori, mostrando sia i valori più elevati di *F1-score medio* e *ponderato*, sia i valori di deviazione standard più bassi tra le varianti testate. Per questo motivo è stato deciso di raffinare ulteriormente il *prompt 1*, generando altre cinque varianti con l'obiettivo di migliorare le performance. Dai risultati, che saranno presentati in maggior dettaglio nel capitolo *5. Risultati*, emerge che il prompt con performance migliori è rimasto il numero 1.

Monolabel zero-shot

Data la difficoltà, nella modalità *multilabel*, di migliorare le performance delle singole etichette in modo indipendente senza influenzare negativamente le altre, è stata introdotta la sperimentazione tramite la tipologia *monolabel zero-shot*. Questo metodo è caratterizzato dalla suddivisione del compito in cinque prompt separati e indipendenti, uno per ciascuna etichetta da classificare.

Rispetto al prompt *multilabel* — che richiede una sola esecuzione per frase — la tipologia *monolabel* presenta uno svantaggio significativo in termini di costi computazionali e di tempo di elaborazione. L'intero processo risulta circa due o tre volte più lento, perché ogni frase deve essere analizzata cinque volte, una per ogni etichetta. Nonostante ciò, questa tipologia è stata esplorata per verificare se l'etichettatura di una singola classe alla volta potesse portare a un incremento delle prestazioni, essendo un compito meno complesso rispetto alla classificazione simultanea di cinque etichette.

La sperimentazione è iniziata ricreando il prompt *multilabel zero-shot* più performante e adattandolo alla modalità *monolabel*. Tuttavia, i risultati ottenuti si sono rivelati inconsistenti e inferiori alle aspettative, per questo motivo sono stati sviluppati quattro nuovi prompt, ciascuno con uno stile diverso. Successivamente, Seguendo una metodologia analoga a quella adottata nella fase precedente, sono stati creati ulteriori cinque prompt, con l'obiettivo di migliorare il *Prompt 5* che, inizialmente, aveva mostrato le performance migliori.

Dai risultati, presentati nel capitolo *5. Risultati*, emerge che il prompt con performance migliori è il numero 10.

Monolabel few-shot

Successivamente, è stato deciso di sperimentare la tipologia *monolabel few-shot*, aggiungendo esempi guidati all'interno del prompt, al fine di osservare un potenziale miglioramento di performance. In totale sono stati testati 12 prompt.

Il corpo del testo utilizzato nel prompt non è stato modificato durante questa sperimentazione, ma è stato riutilizzato il testo del *prompt monolabel few-shot* più performante. Questa decisione è stata ammessa dal fatto che la tipologia di prompt è la stessa; quindi le performance saranno consistenti tra *zero-shot* e *few-shot*, poi dettata dalla volontà di osservare l'effetto delle singole selezioni e variazioni negli esempi.

Le modifiche apportate agli esempi nel corso della sperimentazione hanno riguardato:

- Numero di esempi forniti: 2, 4, 6 o 8 esempi per ciascun'etichetta.
- Modifica degli esempi utilizzati o sostituzione con esempi alternativi.
- Semplificazione sintattica degli esempi complessi e già usati in prompt precedenti.
- Modifica dei parametri del funzionamento del modello.

In particolare, gli esempi sono stati ricavati da un dataset di articoli pubblicati nel 2023 su *ParmaToday*, scelti perché stilisticamente simili a quelli del dataset sperimentale e non inclusi nello stesso, che copre un range di articoli dal 2016 al 2022, al fine di evitare un *data leakage*.

In particolare:

- I prompt 1, 6 e 9 sono composti da 2 esempi per etichetta.

  Il Prompt 6 riprende gli stessi esempi del Prompt 2, ma in una versione semplificata per verificare se una riduzione di lunghezza e della complessità potesse migliorare le performance. Il Prompt 9 utilizza una nuova selezione di esempi, già in forma semplificata.

- Prompt 2, 7 e 10 contengono quattro esempi per classe.

  Il Prompt 7 deriva dal Prompt 3, con frasi semplificate, mentre il Prompt 10 introduce nuovi esempi semplificati.

- Prompt 4 e 5 sono basati sulla struttura del Prompt 2 a quattro esempi. Sono stati creati per testare l'effetto della variazione dei parametri di generazione del modello.

  Nel prompt 4 sono stati impostati temperature = 0.6, top_p = 0.2 e top_k = 40, mentre nel prompt 5 temperature = 0.8, top_p = 0.4 e top_k = 40. Rispetto ai parametri di default (temperature = 0.7, top_p = 0.3), queste modifiche hanno portato a leggeri miglioramenti rispetto al prompt 3, quindi è stato deciso di adottare i parametri del prompt 4 come nuovi valori di riferimento per le prove successive.

- I prompt 3, 8 e 11 utilizzano 6 esempi per etichetta.

  Il Prompt 8 è la versione semplificata del Prompt 3, mentre il Prompt 11 introduce altri nuovi esempi.

- Il prompt 12 è l'unico composto da otto esempi per classe, è stato inserito allo scopo di valutare l'effetto dell'aumento del numero di esempi sulle performance complessive.

Dai risultati, che saranno presentati in dettaglio nel capitolo *5. Risultati*, emerge che il prompt con performance migliori è il 9.

Multilabel few-shot

Per completare il processo di preselezione dei prompt, è stato testato il metodo *multilabel few-shot* per un totale di 11 iterazioni. Come nel caso precedente, le variazioni riguardano principalmente gli esempi, con modifiche relative a:

- Modifica del numero di esempi forniti: 4, 6 oppure 8.
- Scelta degli esempi utilizzati.
- Semplificazione linguistica degli esempi testati in precedenza.

In particolare:

- Prompt 1, 2 e 6 includono 4 esempi.

  Nei Prompt 1 e 2 varia la selezione degli esempi, mentre il Prompt 6 utilizza una versione semplificata degli esempi del Prompt 2.

- Prompt 3, 4, 7 e 9 sono composti da 6 esempi.

  I prompt 3 e 4 estendono rispettivamente i prompt 1 e 2 aggiungendo due esempi. Il prompt 7 riprende il 4 applicando la semplificazione delle frasi, mentre il 9 introduce esempi nuovi e semplificati.

- I prompt 5, 8, 10 e 11 sono composti da 8 esempi.

  Il prompt 5 estende il prompt 3 aggiungendo due esempi. Il prompt 8 semplifica gli esempi del 5. Il prompt 10 si basa su esempi atomici, ciascuno focalizzato su una singola etichetta (eccetto un esempio multi-label). Il prompt 11 combina esempi già utilizzati e nuovi per migliorare le etichette meno performanti. La strategia di utilizzare esempi riguardanti una singola etichetta nei prompt 10 e 11 si basa sull'ipotesi che esempi chiari e meno ambigui possano migliorare la capacità di discriminazione del modello.

Dai risultati, che verranno presentati successivamente, emerge che i prompt con performance migliori sono il numero 2 e il numero 11.

4.1.2. Selezione del prompt finale

Per garantire che la scelta del prompt ottimale fosse coerente con le condizioni operative reali, è stato ritenuto necessario rieseguire una parte dei prompt selezionati in precedenza utilizzando lo stesso ambiente di lavoro previsto per le analisi finali. A questo scopo, è stato impiegato il modello *Qwen3 – 4B Instruct*, eseguito in uno script dedicato sul *cluster HPC*, e applicato al dataset completo composto da 900 istanze.

In totale sono stati analizzati otto prompt. Quattro di questi corrispondono ai migliori per ciascuna tipologia individuata nella fase di preselezione. Gli altri quattro includono i prompt *monolabel few-shot* 6, 7 e 8, selezionati per studiare l'effetto del numero di esempi sulle performance, e il prompt *multilabel few-shot* 2, testato per verificare se le differenze osservate in precedenza rispetto al *prompt 11* si sarebbero confermate anche nel nuovo ambiente di calcolo.

Dai risultati, presentati in maggior dettaglio nel capitolo *5. Risultati*, emerge che il *prompt 9* è quello con performance migliori.

Script HPC utilizzato

Come descritto nei capitoli precedenti, gli esperimenti sono stati eseguiti sul *cluster HPC* dell'Università di Parma.

```
#!/bin/sh
#SBATCH -J Qwen_analysis_%a  ## Nome del lavoro job
#SBATCH -o ./Qwen_analysis_%a_%j.txt  ##Nome del file di log
#SBATCH --partition=gpu
#SBATCH --qos=gpu
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4  ## Numero di core CPU
#SBATCH --gres=gpu:1  ## GPU usata e il numero
#SBATCH --time=0-23:59:59  ## Tempo limite
#SBATCH --mem=30G  ## RAM necessaria in Gb
#SBATCH --array=0-7  ## Array per eseguire 8 script in
parallelo
module load miniconda3
module load gnu7
source "$CONDA_PREFIX/etc/profile.d/conda.sh"
conda activate tesi  ## Ambiente conda

SCRIPTS=(  ## Array di script Python da eseguire
    "analisi_monolabel_few6.py"
    "analisi_monolabel_few7.py"
    "analisi_monolabel_few8.py"
    "analisi_monolabel_few9.py"
    "analisi_monolabel10.py"
    "analisi_multilabel_few2.py"
    "analisi_multilabel1.py"
    "analisi_multilabel11_few.py")
## Esegui lo script corrispondente all'indice dell'array
python ${SCRIPTS[$SLURM_ARRAY_TASK_ID]}
conda deactivate
```

Figura 4: Script usato per lanciare le analisi tramite il comando sbatch nomefile.sh

Per l'esecuzione degli esperimenti di inferenza al fine di svolgere la classificazione binaria, tramite il modello *Qwen3 – 4B Instruct*, sono state allocate le seguenti risorse della partizione GPU del *cluster*. Ogni *job* ha utilizzato una *GPU NVIDIA Tesla P100 PCIe*, 4 core CPU, 30GB di memoria RAM, con un tempo massimo di esecuzione di 23 ore e 59 minuti.

Per ridurre significativamente i tempi di elaborazione sono stati lanciati in parallelo su più script di etichettatura. Il sistema di scheduling SLURM ha gestito automaticamente l'allocazione delle risorse e la distribuzione dei job sui nodi disponibili, permettendo l'esecuzione simultanea delle diverse istanze e ottimizzando l'utilizzo delle risorse computazionali dell'infrastruttura HPC.

4.2. Confronto con OpenData

Il seguente capitolo descrive la pipeline di *data mining* appositamente creata per questa sperimentazione e la sua implementazione. Tale pipeline, inizia dall'ottenimento del dato grezzo, ovvero l'articolo di giornale, e termina con il dato rifinito, ovvero latitudine e longitudine dell'incidente stradale combinato con un indice di pericolosità stradale. Infine, descrive il processo di rifinitura, analisi e visualizzazione dei dati ottenuti. I passi principali sono illustrati in *Figura 4.10* e sono approfonditi nel resto del capitolo.

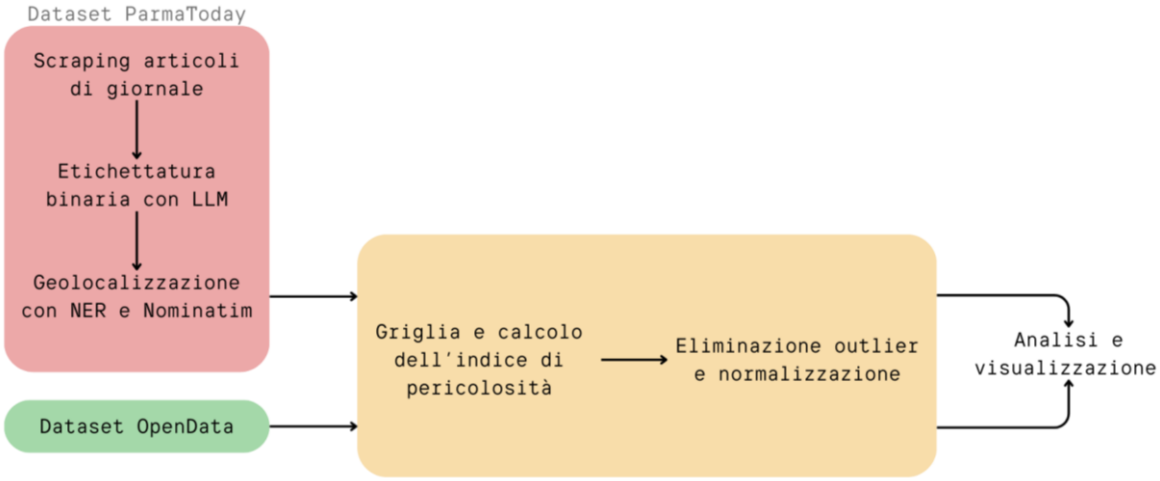

Figura 4.10: Pipeline di esecuzione dal dato grezzo all'analisi e visualizzazione dei dati.

4.2.1. Creazione e preparazione del dataset

Scraping degli articoli di giornale

Per la costruzione del dataset sperimentale sono stati raccolti articoli relativi a incidenti stradali dal quotidiano online *ParmaToday*, limitando la raccolta agli articoli pubblicati nel periodo 2016-2022 per garantire la coerenza temporale con il dataset di riferimento.

Il processo di *scraping* ha sfruttato la funzionalità di ricerca offerta dal sito web, selezionando gli articoli presenti nella sezione di cronaca dedicata agli incidenti stradali. Lo script è stato sviluppato in Python impiegando le librerie *requests* e *BeautifulSoup4* per le richieste *HTTP* delle pagine e l'estrazione dei contenuti.

Per ciascuna pagina sono stati recuperati i link agli articoli, dai quali sono stati estratti il titolo, la data di pubblicazione e il corpo del testo. I dati ottenuti sono stati poi salvati in un file di testo seguendo la seguente struttura:

- #URL
- Titolo
- ~ Data
- Testo
- --- .

Questa formattazione consente di identificare facilmente ciascun elemento mediante espressioni regolari. Lo pseudocodice dello script di *scraping* è riportato in *Figura 4.11*.

```
1. CONFIGURAZIONE INIZIALE:
 - base_url
 - output_file ← "articoli_parma.txt"

2. PER OGNI numero di pagina da DA 1 A end_page:
 a. Costruisci URL della pagina come base_url + page_number

 b. Ottieni lista URL degli articoli:
   - Richiesta GET alla pagina
   - Trova tutti i link <a> con href e filtra quelli
     generici o che non iniziano con base_url
   - Se link valido e non duplicato: aggiungi a lista

 c. Per ogni elemento della lista articoli:
     - Richiesta GET all'articolo
     - Estrai il titolo (primo <h1>)
       Altrimenti: titolo ← "Titolo non trovato"
     - Estrai la data (giorno/mese/anno)
       Altrimenti: data ← "Data non trovata"
     - Estrai il testo cercando tag <article> e trovando
       tutti i <p> al suo interno e concatenandoli
       Altrimenti: contenuto ← "Contenuto non trovato"

3. Salva il file ogni 5 articoli e aspetta 1-2s per essere
gentile al sito.
```

Figura 4.11: Pseudocodice relativo allo script di scraping del giornale online ParmaToday.

Preparazione del dataset

Effettuato lo *scraping*, il testo di ogni articolo è stato suddiviso in frasi e inserito in un file *CSV* contenente: l'ID dell'articolo, la frase, e le cinque etichette inizializzate a zero (localizzazione, tempo, grave, moto, utenti deboli).

Questa fase è risultata necessaria per ottenere i dati in un formato compatibile con lo script di etichettatura.

4.2.2. Etichettatura e raffinazione del dataset

L'etichettatura è stata effettuata tramite il modello *Qwen3 – 4B Instruct* eseguito sull'infrastruttura *HPC* di ateneo. I dettagli implementativi relativi allo script impiegato coincidono con quelli descritti nella fase di selezione dei prompt, dalla quale è stato utilizzato il prompt con le migliori performance.

Una volta ottenute le predizioni dal modello *LLM*, è stato applicato un processo di raffinazione con l'obiettivo di ridurre la ridondanza e garantire coerenza logica tra le etichette assegnate. Tale operazione è stata necessaria perché ogni articolo è stato analizzato frase per frase generando, potenzialmente, multiple occorrenze della stessa etichetta all'interno di un singolo articolo.

Poiché le analisi successive richiedevano esclusivamente le etichette localizzazione e grave, le altre tre etichette (tempo, moto, utenti deboli) sono state rimosse durante la fase di raffinazione. Per evitare duplicazioni della localizzazione, è stata mantenuta soltanto la prima frase rappresentativa dell'articolo, ovvero la prima in cui l'etichetta localizzazione assume valore pari a 1. L'implementazione utilizza un buffer che raccoglie progressivamente le frasi appartenenti allo stesso articolo e applica le trasformazioni al momento del cambio dell'ID di articolo, in più, garantisce la corretta gestione dell'ultimo articolo del file.

Poi l'etichetta grave, se presente positivamente in almeno una frase dell'articolo interessato, viene trasferita nell'unica frase mantenuta. Il funzionamento complessivo è illustrato nello script riportato in *Figura 4.12;* per ogni articolo vengono accumulate tutte le frasi in buffer, viene individuata la prima con localizzazione positiva in cui viene propagata l'informazione relativa alla gravità, se presente in almeno una frase dell'articolo.

```
# se almeno una frase ha grave=1, allora diventa True
grave_in_articolo = any(row[3] == 1 for row in buffer)
for row in buffer: #itera su tutte le frasi dell'articolo
  if row[2] == 1: # se localizzazione=1, mantieni la frase
    if grave_in_articolo: # se grave presente,
                          imposta grave=1
      row[3] = 1
    writer.writerow(row) # scrivi in output csv
    break
```

Figura 4.12: Sezione di codice della selezione della prima frase rappresentativa dell'articolo.

4.2.3. Geolocalizzazione

NER

Per l'estrazione automatica delle località citate nelle frasi è stato impiegato il modello *spaCy it_nerIta_trf*, proveniente dalla libreria *spacy*, che riporta un *F1-score* pari a 0.915 secondo la pagina *HuggingFace*. [17]

Durante la sperimentazione sono stati applicati diversi filtri e miglioramenti dell'output del modello, che inizialmente otteneva prestazioni molto basse.

Prima dell'estrazione, è stato applicato un arricchimento contestuale tramite *ReGEx*: nel caso in cui la frase contenesse prefissi stradali come, piazza, strada o piazzale seguiti da un'entità riconosciuta, tali elementi, vengono concatenati per ottenere un toponimo

completo (ad esempio “Strada Argini” invece di “Argini”). Questo intervento ha migliorato sensibilmente la precisione della successiva fase di geocodifica.

Dopo l’estrazione, sono stati applicati filtri di post-processing per ridurre falsi positivi e ridondanze. lo script scarta le entità contenenti la parola "ospedale", non rappresentanti della localizzazione stradale dell'incidente, e tutte le occorrenze di "Parma", ridondanti in quanto tutti gli eventi si riferiscono allo stesso contesto geografico. Infine, un meccanismo di deduplicazione si assicura che ogni entità venga estratta una sola volta per frase.

```
def estrai_luoghi_NER(doc, frase):
  entities = [] #lista di entità trovate
  seen = set()  #set per evitare duplicati
  for ent in doc.ents:
    # per le entità GPE e FAC trovate esclendo 'ospedale'
    if (ent.label_ == "GPE" or ent.label_ == "FAC") and
       "ospedale" not in ent.text.lower():
      # cerca prefissi nella frase
      pattern = re.compile(rf"(strada|piazza|piazzale)\s+{
      re.escape(ent.text)}", re.IGNORECASE)
      # concatena prefisso + entità interessata
      if pattern.search(frase):
        match = pattern.search(frase)
        prefisso = match.group(1)
        entity = f"{prefisso} {ent.text}"
      else:
        entity = ent.text
      if entity not in seen: # evita duplicati
        entities.append(entity)
        seen.add(entity)
  # Rimuovi tutte le occorrenze di 'Parma'
  entities = [e for e in entities if not
              re.fullmatch(r"\s*parma\s*", e, re.IGNORECASE)]
  return entities
```

Figura 4.13: Script per estrarre e filtrare le entità NER.

Nominatim

Le entità estratte tramite *NER* vengono successivamente geo-codificate tramite il servizio *Nominatim*, al fine di ottenere le coordinate geografiche dell’incidente. Ciascuna richiesta, eseguita tramite la libreria *requests*, viene arricchita con il contesto geografico "Parma, Emilia-Romagna, Italia" per migliorare la precisione delle corrispondenze.

Dato che un buon numero di frasi contenevano più di un’entità di localizzazione, è stata implementata una strategia “a cascata” per massimizzare il tasso di successo. Inizialmente, ogni entità viene interrogata singolarmente; se viene trovata una corrispondenza valida le coordinate vengono salvate, altrimenti, si passa all’entità successiva disponibile. Se nessuna entità ha restituito risultati validi, viene effettuata una

query combinata contenente tutte le entità separate da virgola, tentando di sfruttare il contesto complessivo per ottenere una possibile localizzazione.

Per ogni localizzazione individuata con successo, il sistema restituisce: latitudine, longitudine e la descrizione testuale completa della posizione secondo *OpenStreetMap*. Nel dataset finale vengono mantenute esclusivamente le frasi per cui la geocodifica ha esito positivo, garantendo che ogni record contenga coordinate valide e idonee alle analisi successive.

```
def estrai_coordinate_nominatim(entities_str):
  if entities_str:
    # inserisci in lista le entità divise da virgola
    entities = [e.strip() for e in entities_str.split(",")
if e.strip()]
    if not entities: # error handling: lista vuota
      return None, None, None
    for ent in entities: # Prova ogni entità singolarmente
      # prefissi costanti per migliorare accuratezza
      single_query = f"{ent}, Parma, Emilia-Romagna, Italia"
      # chiama HTTP GET request
      lat, lon, pos = get_coordinates(single_query)
      if lat and lon: # se coordinate valide, output
        return lat, lon, pos
    # Se nessuna singola funziona, prova tutte insieme
    combined_query = f"{', '.join(entities)}, Parma,
                     Emilia-Romagna, Italia"
    lat, lon, pos = get_coordinates(combined_query)
    if lat and lon: # controllo se le coordinate sono valide
      return lat, lon, pos
return None, None, None
```

Figura 4.14: Funzione di gestione dell'estrazione di coordinate dalle entità tramite Nominatim.

4.2.4. Scrematura geografica

Siccome il dataset ottenuto dallo *scraping* comprende articoli relativi all'intera provincia di Parma, mentre, il dataset di benchmark copre solamente per l'area relativa al comune, è stato necessario applicare un filtro geografico per mantenere solo gli incidenti effettivamente avvenuti all'interno del comune di Parma.

A questo scopo, tramite la libreria *shapely*, è stato definito manualmente un poligono rappresentativo dei confini comunali utilizzando 25 coordinate che ne delimitano il perimetro, estratte dalla mappa cartografica interattiva ufficiale disponibile online. Poi, per ogni coppia di coordinate prodotto dallo script di geolocalizzazione, viene creato un oggetto punto e verificato se questo ricade all'interno del *Polygon comunale*. [32]

Questo controllo scarta automaticamente i risultati esterni ai confini scelti, consentendo di mantenere solo gli incidenti effettivamente pertinenti all'area di studio, diminuendo, così, il numero di incidenti utilizzabili da 1777 a 612 per il dataset sperimentale.

Da questo punto in poi, la seguente pipeline è stata applicata anche per il dataset *OpenData*, in cui questo filtro geografico ha portato alla rimozione di soli 3 incidenti, riducendo il totale da 5104 a 5101.

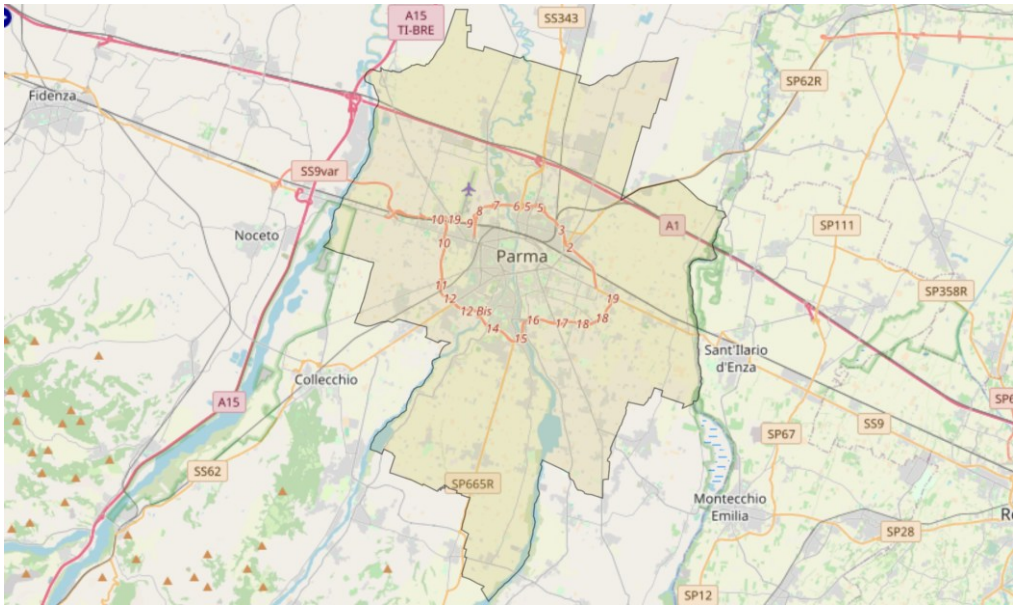

Figura 4.15: Confini del comune di Parma, fonte: OpenStreetMap.

4.2.5. Calcolo dell'indice di pericolosità stradale in griglia

Per consentire un confronto diretto tra il dataset sperimentale e quello di riferimento, su entrambi è stato applicato lo stesso processo di calcolo dell'indice di pericolosità stradale.

È stata, quindi, generata una griglia regolare di punti equidistanti che segmenta l'area di studio in partizioni omogenee. Attorno a ciascun punto è stata costruita una cella quadrata di dimensione fissa, ottenuta tramite *bounding box*, utilizzando la libreria *geopy* per il calcolo delle distanze geografiche. La distanza scelta come passo della griglia e come lato del quadrato è di 0.5 km. L'utilizzo della medesima griglia per entrambi i dataset consente un confronto diretto punto-punto, discusso in seguito.

La scelta del passo di 0.5 km deriva da un compromesso tra diversi fattori tra cui:

- Un passo troppo piccolo sarebbe stato incompatibile con la scarsa precisione delle informazioni contenute negli articoli giornalistici. La maggior parte degli articoli non riporta il punto specifico di accadimento dell'incidente, ma solo la via, che può estendersi per diversi chilometri (ad esempio via Emilia Est o via Emilia Ovest).
- Un passo troppo grande avrebbe annullato l'utilità reale della mappa di rischio, in particolare nelle aree centrali della città, più dense in termini di viabilità e traffico.

Per il calcolo dell'indice di pericolosità stradale, tramite la libreria *numpy*, vengono considerati tutti e soli gli incidenti che ricadono all'interno della rispettiva *bounding box*. L'indice è il risultato della somma del peso discreto associato a ciascun evento:

- Viene assegnato un peso pari a 1.0 per gli incidenti gravi, quindi con l'etichetta grave = 1.
- Viene dato un valore di 0.5 per gli incidenti non gravi.

Se una cella non contiene incidenti, il relativo indice di pericolosità è impostato a 0.0. L'indice finale della cella è, quindi, la somma dei pesi degli incidenti in essa contenuti, rappresentando una misura congiunta di frequenza e severità degli eventi.

4.2.6. Eliminazione degli outlier e normalizzazione

In questo script vengono utilizzate le librerie *pandas* per la lettura e la manipolazione dei dati, *matplotlib* per la creazione di grafici e istogrammi, e *numpy* per l'elaborazione numerica e il calcolo di statistiche come i quartili e il range interquartile.

Tipologia di distribuzione dei dati

Prima di procedere alla raffinazione del dataset e all'analisi, è stato necessario caratterizzare la distribuzione dell'indice di pericolosità stradale. Per farlo sono stati prodotti istogrammi con passo 0.5, sia includendo e sia escludendo i punti per cui l'indice è pari a zero.

La *Figura 4.16* mostra la distribuzione della frequenza relativa dei valori di rischio per entrambi i dataset, includendo anche gli zeri. In questa configurazione, la distribuzione risulta fortemente alterata, perché per entrambi i dataset oltre metà dei punti assume valore zero.

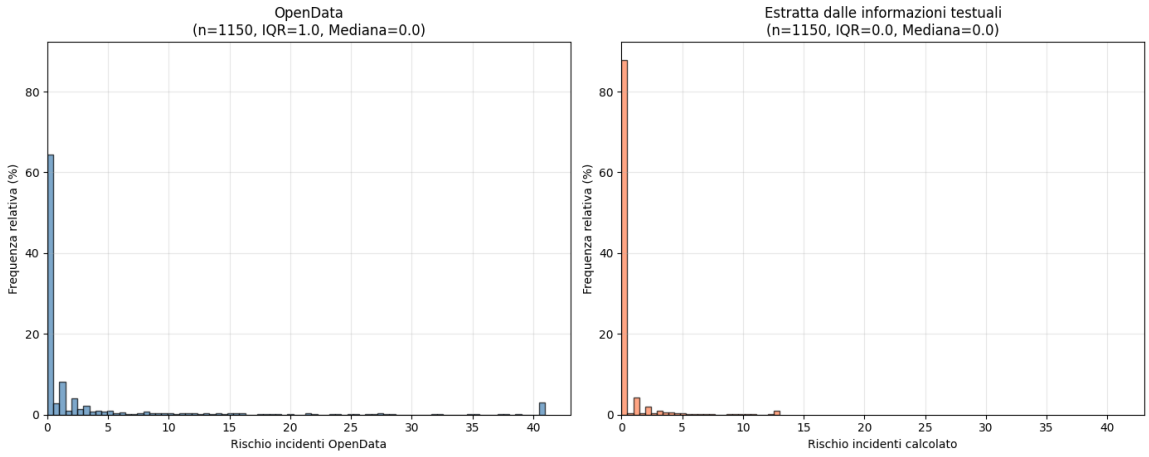

Figura 4.16: Istogramma della frequenza di incidenti percentuale per indice di rischio inclusi i valori a zero.

Per questo motivo è stata esaminata nuovamente la distribuzione escludendo i punti con indice pari a zero, come mostrato in *Figura 4.17* e *Figura 4.18.*

Entambe le distribuzioni filtrate presentano un'asimmetria concentrata nei valori dell'indice di rischio bassi. È inoltre presente un insieme di picchi isolati agli estremi delle distribuzioni. Nonostante tali similitudini, gli istogrammi risultano complessivamente diversi, come evidenziato in *Tabella 4.6*.

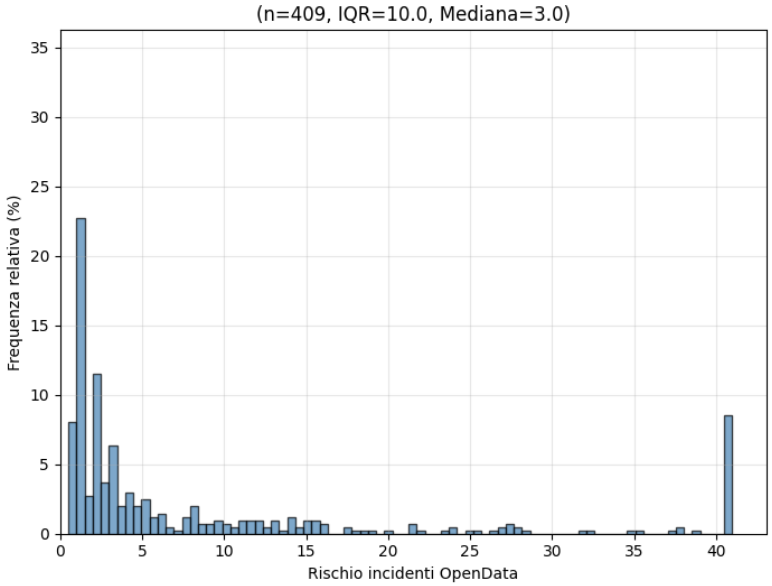

Figura 4.17: Istogramma della frequenza di incidenti percentuale per indice di rischio, esclusi i valori pari a zero, dati estratti da dataset di riferimento.

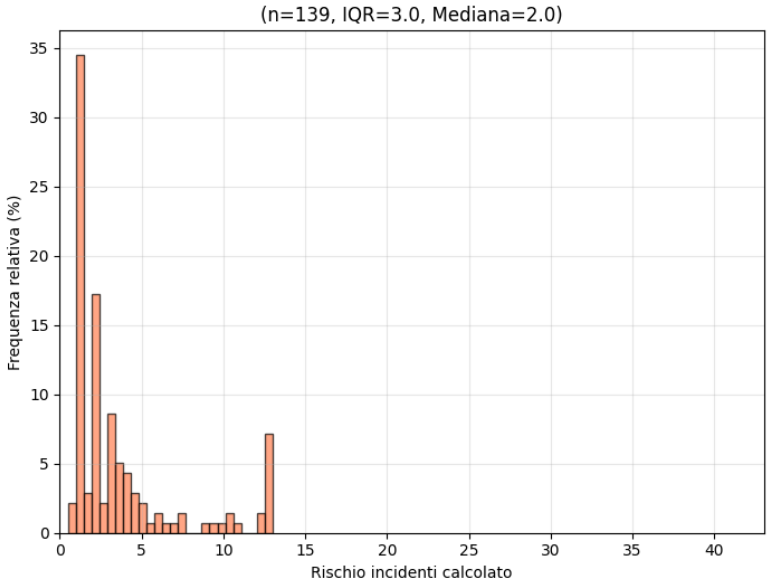

Figura 4.18: Istogramma della frequenza di incidenti percentuale per indice di rischio, esclusi i valori pari a zero, dati estratti da dataset sperimentale.

Eliminazione degli outlier

Dato il tipo di distribuzione osservata, è stato implementato un sistema di rilevamento e gestione degli outlier basato sul metodo *IQR* (Interquartile Range) di *Tukey*, applicato solo alle celle con indice di pericolosità maggiore di zero.

Il metodo calcola i tre quartili (Q1, Q2, Q3) della distribuzione e definisce l'IQR come:

$$IQR = \mathrm{Q3} - \mathrm{Q1}$$

La soglia di outlier è definita come:

$$Soglia = \mathrm{Q3} + \mathrm{k} \cdot \mathrm{IQR}, \quad \mathrm{k} = 3.0$$

Il parametro k scelto rappresenta un criterio più conservativo rispetto a quello classico di 1.5. Questa scelta permette di mantenere casi plausibili di elevata pericolosità, limitando i valori anomali attribuibili a errori di geolocalizzazione o aggregazione. Tutti i valori superiori alla soglia vengono sostituiti con la soglia stessa, preservando l'informazione spaziale della cella ma riducendo l'impatto di anomalie statistiche.

Metrica	OpenData	Esperimento
N incidenti nella provincia	5104	1436
N incidenti nel comune	5101	612
Mediana	3	2
Range	[0, 130.5]	[0, 45.0]
IQR	10	3
Soglia outlier	13	41

Tabella 4.6: metriche in confronto tra i due dataset.

Il dataset di riferimento presenta una maggiore densità di incidenti, con valori di gravità tendenzialmente più elevati. Al contrario, il dataset sperimentale mostra una distribuzione caratterizzata da un range limitato. Questa differenza riflette la natura sbilanciata delle due fonti: i dati ufficiali registrano sistematicamente tutti gli incidenti denunciati, mentre gli articoli giornalistici selezionano pochi eventi con maggiore rilevanza mediatica.

Normalizzazione

Per compensare lo sbilanciamento nel numero di istanze e rendere confrontabili le magnitudini dei due dataset, è stata applicata una *normalizzazione Min-Max*, eseguita separatamente per ciascun dataset. Questo metodo consente di riscalare i valori nell'intervallo [0,1], eliminando l'effetto dei range differenti e ampi senza alterare le relazioni interne tra i dati. [33]

La scelta della *normalizzazione Min-Max* è motivata dall'esigenza di porre le due misurazioni sulla stessa scala, rendendo più chiaro e coerente il confronto successivo delle distribuzioni e delle correlazioni, senza alterare i risultati delle analisi.

La formula usata è per ogni record di entrambi i dataset è:

$$valore\ normalizzato_i = \frac{valore_i - minimo}{massimo - minimo}$$

I risultati della normalizzazione sulla distribuzione sono mostrati in *Figura 4.19* e *Figura 4.20*.

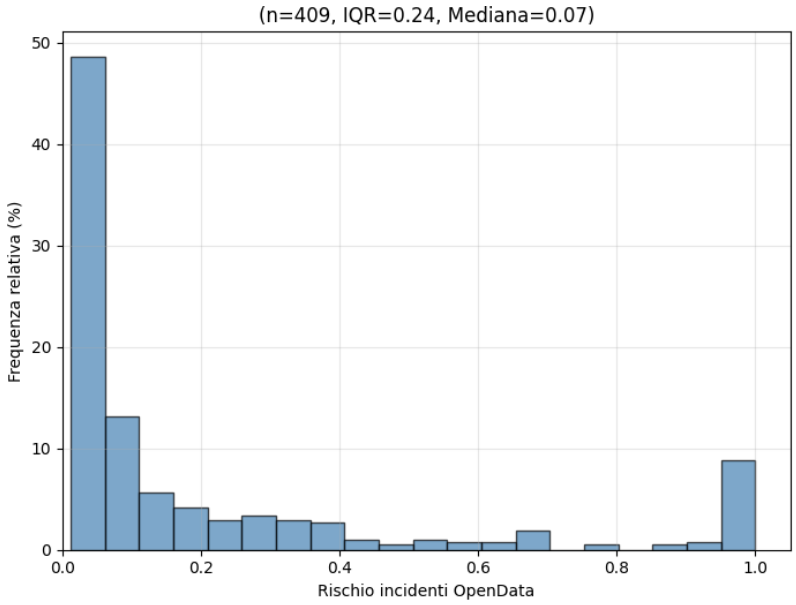

Figura 4.19: Istogramma della frequenza di incidenti percentuale per indice di rischio, esclusi i valori pari a zero, dopo la normalizzazione Min-Max, dati provenienti da dataset di riferimento.

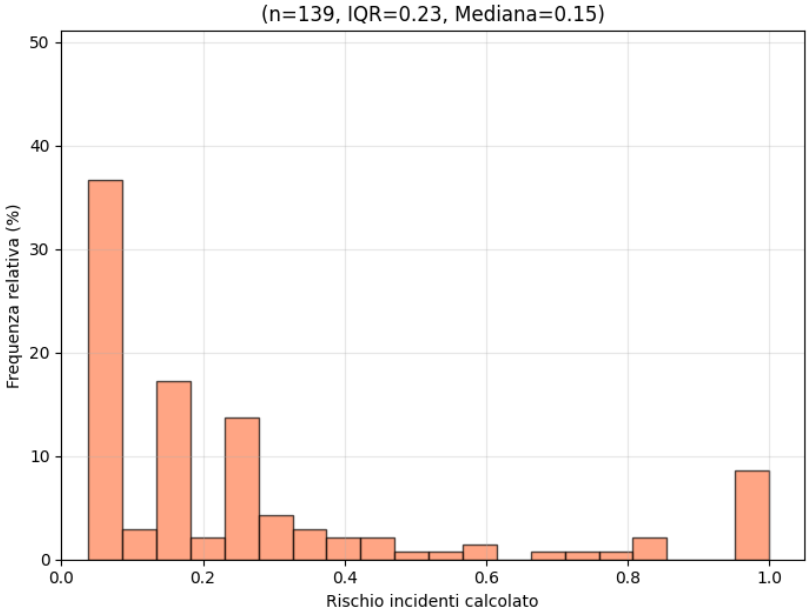

Figura 4.20: Istogramma della frequenza di incidenti percentuale per indice di rischio, esclusi i valori pari a zero, dopo la normalizzazione Min-Max, dati provenienti da dataset sperimentale.

4.2.7. Matching punto-punto dei due dataset per confrontare la gravità

Per confrontare quantitativamente i due metodi di misura è stato implementato un sistema di matching punto-punto tra i dataset normalizzati. Le coppie vengono abbinate verificando che le coordinate dei punti associati alle *bounding box* siano congruenti per i valori analizzati.

L'output è inserito in un file *CSV* in cui vengono riportate: coordinate, valori dell'indice di pericolosità del metodo che misura a partire dal dataset di riferimento e quelli ottenuti dal dataset sperimentale. Questo formato consente di effettuare l'analisi in modo più semplice.

4.2.8. Analisi e visualizzazione

Analisi grafica e quantitativa

Per valutare la correlazione lineare tra i due metodi di misurazione è stato calcolato l'indice *Pearson r*, accompagnato dal *p-value* per verificarne la significatività statistica. Le librerie utilizzate per l'analisi grafica sono *matplotlib* per la creazione dei grafici e *scipy.stats* al fine di calcolare l'indice di correlazione lineare.

Gli strumenti grafici utilizzati sono stati:

- *Scatter plot*: la gravità rilevata a partire dal dataset di riferimento è riportata sull'asse delle ascisse, mentre la gravità calcolata tramite la pipeline sperimentale è riportata sull'asse delle ordinate. Sui grafici è stata aggiunta la linea di corrispondenza perfetta per agevolare l'interpretazione visiva dei dati.
- *Bland-Altman plot*: ogni punto rappresenta la media dei valori dei due metodi sull'asse delle ascisse e la differenza tra i valori dell'indice di rischio dei dataset benchmark e sperimentale sull'asse delle ordinate. Questo strumento permette di valutare l'accordo tra i due metodi e calcolare il *Bias*.

Tutti i grafici sono stati generati mediante script Python dedicati, utilizzando la libreria matplotlib.

Heatmap

Le *heatmap* sono state utilizzate come strumento per la visualizzazione geografica dei dati e per ottenere una rappresentazione intuitiva dell'indice di pericolosità stradale. La creazione delle mappe è stata effettuata tramite la libreria *Folium*, generando mappe in formato *HTML* interattivo. Sono state generate diverse tipologie di *heatmap*:

- Indice di pericolosità stradale calcolato per ciascun punto della griglia, separatamente per il dataset *OpenData* e per quello sperimentale.
- Confronto tra i due metodi, evidenziando le aree di sovrastima e sottostima dell'indice di pericolosità stradale.

5. Risultati

5.1. Risultati Selezione dei prompt

5.1.1. Multilabel zero-shot

La prima fase di sperimentazione ha riguardato la configurazione *multilabel zero-shot*, la più semplice tra le modalità analizzate, di cui sono state testate 10 varianti. Da questa fase preliminare è emerso che il prompt più efficace è il numero 1, caratterizzato da uno stile informale con domande dirette.

Sebbene siano state testate 5 varianti del Prompt 1 e sia stato possibile incrementare le prestazioni relative ad alcune etichette specifiche, non è stato possibile superare le performance medie complessive del *Prompt 1*.

Come riportato in *Tabella 5.1*, che presenta i risultati medi ottenuti nella sperimentazione con la modalità *multilabel zero-shot*, si osserva una bassa variabilità delle performance per la maggior parte dei prompt testati. L'unica eccezione è rappresentata dal *Prompt 2*, che è stato escluso in quanto caratterizzato da una deviazione standard significativamente superiore rispetto alla media. Per tutti gli altri prompt, il *coefficiente di variazione* (CV%) risulta inferiore al 5%, indicando una buona stabilità dei risultati.

Questo comportamento è stato confermato durante tutto il processo sperimentale e suggerisce che l'impiego del *LLM* per questo task di etichettatura binaria possa garantire una consistenza elevata dei risultati.

Prompt	Media F1 medio	Media F1 ponderato	CV% F1 medio	CV% F1 ponderato
1	0.66 ± 0.01	0.74 ± 0.01	1.2	1.0
2	0.53 ± 0.06	0.63 ± 0.07	11.5	11.0
3	0.55 ± 0.01	0.61 ± 0.01	1.0	0.9
4	0.64 ± 0.01	0.69 ± 0.01	0.8	0.8
5	0.63 ± 0.02	0.69 ± 0.01	2.9	0.8
6	0.64 ± 0.01	0.69 ± 0.01	2.2	2.0
7	0.57 ± 0.02	0.64 ± 0.01	3.1	2.2
8	0.61 ± 0.01	0.69 ± 0.02	2.4	3.2
9	0.64 ± 0.01	0.71 ± 0.02	2.3	3.1
10	0.63 ± 0.01	0.71 ± 0.01	2.2	1.9

Tabella 5.1: KPI medi relativi ai prompt multilabel zero-shot.

Per rendere più immediata l'analisi comparativa è stato, inoltre, realizzato il grafico delle quattro varianti di prompt con le performance migliori, mostrato in *Figura 5.1*.

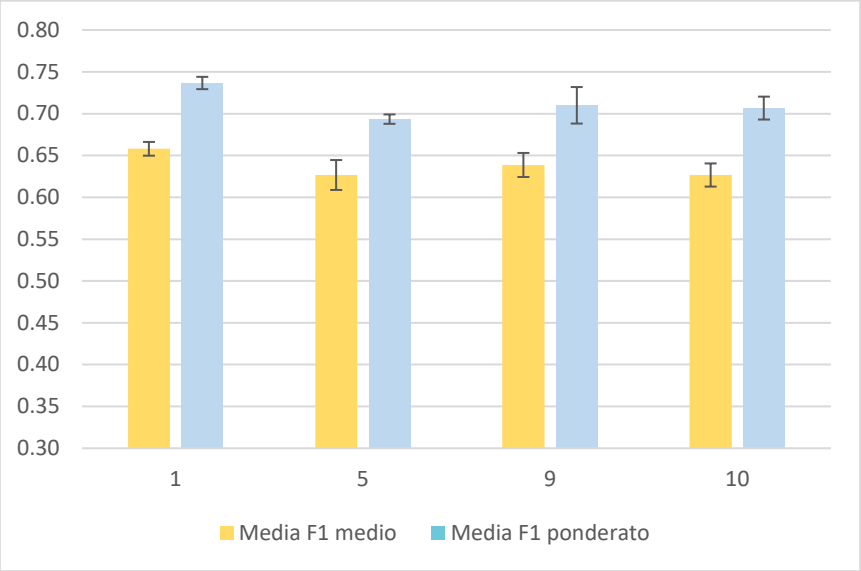

Figura 5.1: Performance medie dei quattro prompt migliori.

5.1.2. Monolabel zero-shot

Per superare i limiti della classificazione *multilabel* è stata introdotta la modalità *monolabel zero-shot*; che prevede la creazione di cinque prompt indipendenti, uno per ciascuna etichetta. Ciò ha aumentato significativamente il costo computazionale — ogni frase viene analizzata cinque volte — ma ha semplificato il task di classificazione. Utilizzando un approccio analogo a quello illustrato precedentemente, le iterazioni hanno portato all'individuazione del *prompt 10* come configurazione più performante per la modalità *monolabel zero-shot*.

La *Tabella 5.2* riporta i risultati ottenuti per tutti i 10 prompt testati in questa modalità.

Prompt	Media F1 medio	Media F1 ponderato	CV% F1 medio	CV% F1 ponderato
1	0.59 ± 0.04	0.71 ± 0.02	7.5	3.4
2	0.55 ± 0.03	0.75 ± 0.05	5.8	6.6
3	0.55 ± 0.02	0.73 ± 0.04	3.1	4.8
4	0.57 ± 0.02	0.71 ± 0.04	3.0	4.9
5	0.63 ± 0.03	0.72 ± 0.05	5.3	7.5
6	0.64 ± 0.04	0.77 ± 0.05	5.6	6.7
7	0.59 ± 0.05	0.73 ± 0.01	7.9	1.7
8	0.57 ± 0.01	0.77 ± 0.01	2.1	1.4
9	0.59 ± 0.02	0.77 ± 0.02	3.1	2.5
10	0.65 ± 0.02	0.79 ± 0.01	3.5	1.3

Tabella 5.2: KPI medi relativi ai prompt monolabel zero-shot.

In conclusione, modificando sia il lessico che la formulazione delle domande, è stato possibile migliorare le performance medie del *Prompt 5* e, soprattutto, ridurre significativamente il *CV%* per entrambi i *KPI*, ottenendo un risultato migliore grazie all'applicazione dei principi di *prompt engineering*.

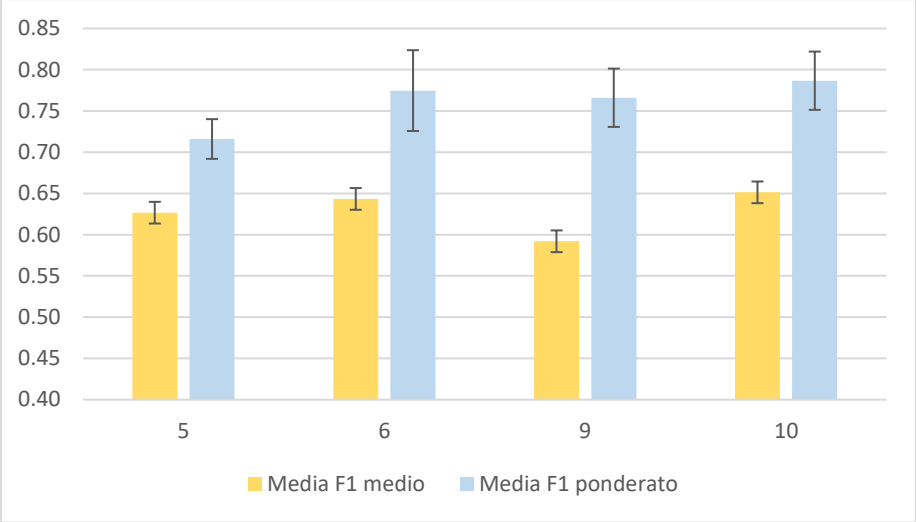

Figura 5.2: Performance medie dei quattro prompt migliori.

5.1.3. Monolabel few-shot

La terza fase ha introdotto esempi con soluzione all'interno dei prompt, modalità *few-shot*, per verificare se l'aggiunta di contesto esplicito potesse migliorare le performance. In totale sono stati testati 12 prompt, variando diversi parametri come: numero di esempi, selezione delle frasi, complessità linguistica e parametri di generazione del modello. Gli esempi sono stati ricavati da articoli esclusi dal dataset principale al fine di evitare il *data leakage*.

I risultati della sperimentazione *monolabel few-shot* sono riportati in *Tabella 5.3*. Come osservabile, il *Prompt 9* — composto da due esempi semplificati per etichetta — ha ottenuto le migliori performance; tuttavia, tali risultati non differiscono in modo sostanziale da quelli dei *Prompt 6* e *7*.

Come illustrato nella *Figura 5.3*, che riporta le performance dei *prompt 9-12* in funzione del numero di esempi forniti, si osserva una chiara tendenza di diminuzione delle prestazioni all'aumentare del numero di esempi inclusi nel prompt. Quindi, rivelando come l'aggiunta di esempi non abbia portato a un miglioramento delle performance. Questo risultato può essere attribuito a diversi fattori, tra cui l'aumento lunghezza del prompt e la selezione degli esempi, irrilevanti o di difficile interpretazione, che potrebbero aver aumentato la confusione del modello in quanto di dimensioni contenute. Per questo motivo è stato deciso di testare ugualmente le varianti a 4 e 6 esempi nella fase successiva, che prevede l'utilizzo di un modello più grande (4B) e per verificare se l'aumento di capacità potesse migliorare le performance.

Prompt	Media F1 medio	Media F1 ponderato	CV% F1 medio	CV% F1 ponderato
1	0.60 ± 0.01	0.77 ± 0.01	1.7	1.3
2	0.58 ± 0.01	0.75 ± 0.01	1.7	1.3
3	0.58 ± 0.01	0.71 ± 0.01	1.7	1.4
4	0.61 ± 0.01	0.77 ± 0.01	1.6	1.3
5	0.61 ± 0.01	0.77 ± 0.01	1.6	1.3
6	0.66 ± 0.01	0.80 ± 0.01	1.5	1.2
7	0.66 ± 0.01	0.80 ± 0.01	1.5	1.2
8	0.62 ± 0.01	0.78 ± 0.01	1.6	1.3
9	0.66 ± 0.01	0.81 ± 0.01	1.5	1.2
10	0.64 ± 0.02	0.81 ± 0.01	3.1	1.2
11	0.62 ± 0.01	0.78 ± 0.01	1.6	1.3
12	0.59 ± 0.01	0.76 ± 0.01	1.7	1.3

Tabella 5.3: KPI medi relativi ai prompt monolabel few-shot.

Un comportamento analogo è stato osservato anche per i *Prompt 6–8*, suggerendo che, per questo task, con questa metodologia e con il modello adottato, esista un punto ottimale nel numero di esempi, collocato tra 2 e 4.

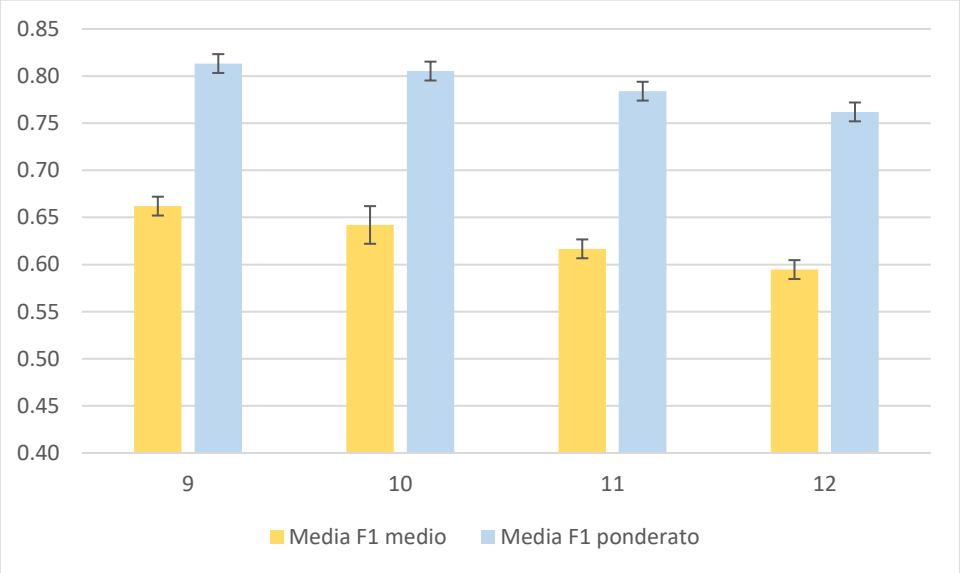

Figura 5.3: Performance medie del prompt migliore all'aumento del numero di esempi (2, 4, 6, 8).

5.1.4. Multilabel few-shot

Per completare la preselezione, è stata valutata anche la modalità *multilabel few-shot*, testando 11 prompt costruiti variando il numero di esempi (4, 6 oppure 8), la selezione delle frasi e il livello di semplificazione linguistica. Sono stati, inoltre, introdotti esempi focalizzati su singole etichette per ridurre ambiguità e migliorare la precisione.

Un'osservazione interessante, riguarda l'effetto della semplificazione sintattica degli esempi. I *prompt 6, 7* e *8*, che utilizzano versioni semplificate degli esempi dei *prompt 2*, *4* e *5*, hanno mostrato performance medie inferiori rispetto ai corrispondenti prompt con esempi completi. Questo suggerisce che, per il metodo di etichettatura binaria *multilabel*, la completezza delle informazioni è più importante della semplicità dell'esempio.

Per tutti i prompt testati, i valori di *CV%* si sono mantenuti molto bassi, indicando una buona stabilità e riproducibilità dei risultati anche per questa metodologia di prompting.

Prompt	Media F1 medio	Media F1 ponderato	CV% F1 medio	CV% F1 ponderato
1	0.66 ± 0.01	0.75 ± 0.01	1.5	1.3
2	0.67 ± 0.01	0.74 ± 0.01	1.5	1.3
3	0.61 ± 0.01	0.72 ± 0.01	1.6	1.4
4	0.65 ± 0.01	0.73 ± 0.01	1.5	1.4
5	0.62 ± 0.02	0.72 ± 0.02	3.2	2.8
6	0.62 ± 0.01	0.73 ± 0.01	1.6	1.4
7	0.61 ± 0.01	0.69 ± 0.01	1.6	1.4
8	0.62 ± 0.01	0.70 ± 0.01	1.6	1.4
9	0.65 ± 0.01	0.72 ± 0.01	1.5	1.4
10	0.66 ± 0.01	0.74 ± 0.01	1.5	1.3
11	0.67 ± 0.01	0.74 ± 0.01	1.5	1.3

Tabella 5.4: KPI medi relativi ai prompt multilabel few-shot.

Di seguito è stato riportato un grafico a istogramma dei *KPI* relativi ai quattro prompt più performanti tra gli 11 testati. Dall'analisi del grafico, si può notare come le performance dei prompt selezionati, considerando la deviazione standard, siano sostanzialmente identiche. Per questo motivo, nella selezione successiva è stato scelto arbitrariamente di utilizzare i *prompt 2* e *11*.

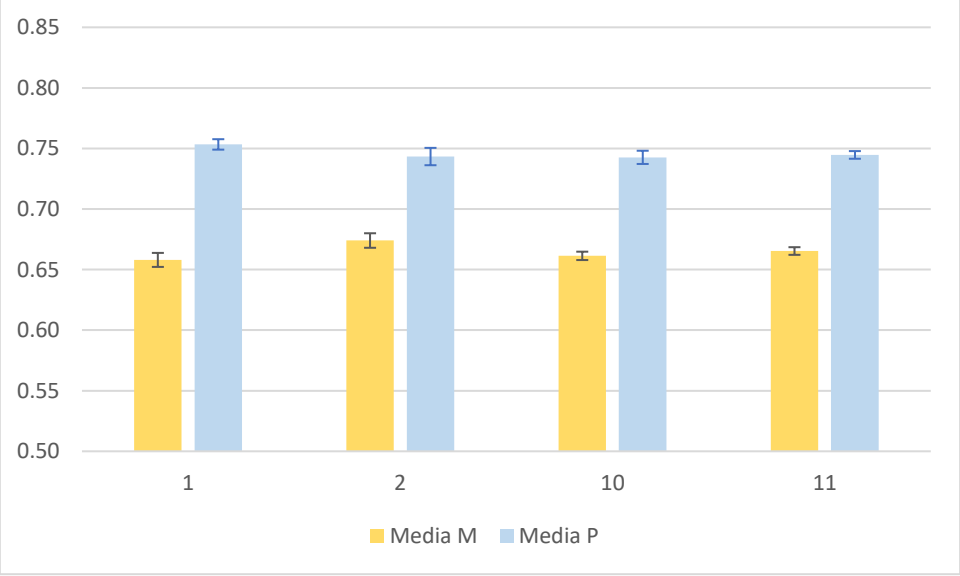

Figura 5.4: Performance medie dei quattro prompt migliori.

5.1.5. Analisi conclusiva

I prompt migliori per ciascuna tipologia sono illustrati in *Figura 5.5*. Dal grafico emerge chiaramente come le diverse strategie di prompting, selezionate attraverso la fase di preselezione, presentino performance molto simili tra loro, pur mostrando un miglioramento più evidente nella modalità monolabel.

In particolare, il parametro che ha beneficiato maggiormente del processo di ottimizzazione è l'*F1-score ponderato* medio, mentre la media dell'F*1-score medio* è rimasto pressoché invariato per tutte le configurazioni di prompt.

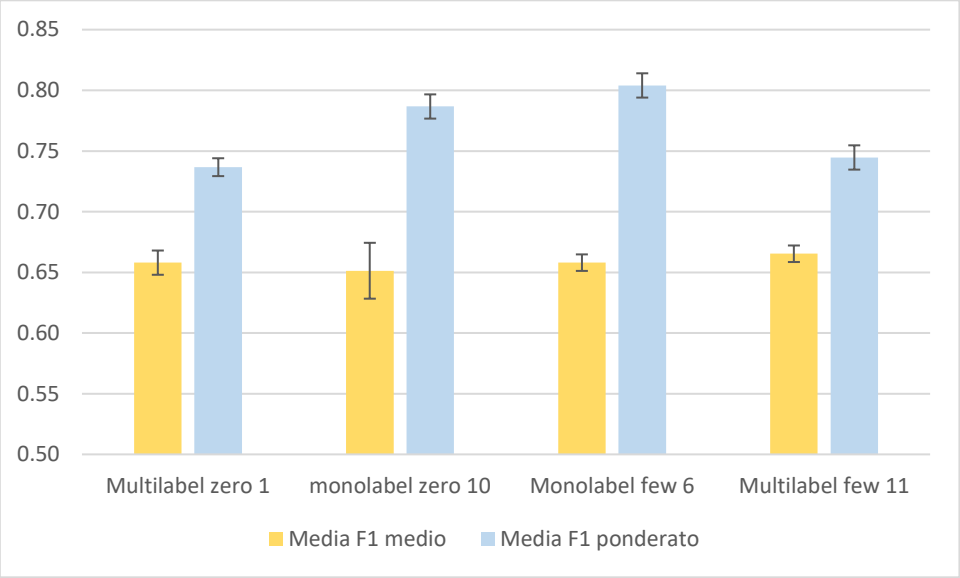

Figura 5.5: Performance medie dei prompt migliori per tipologia.

5.1.6. Risultati selezione finale

Infine, per selezionare il prompt ottimale in condizioni operative reali, una parte dei migliori prompt individuati nelle fasi precedenti è stata rieseguita su *cluster HPC*, utilizzando il modello *Qwen3 - 4B Instruct* e il dataset completo. Sono stati testati otto prompt, includendo sia i migliori delle quattro tipologie, sia ulteriori varianti *monolabel few-shot* dedicate ad analizzare l'impatto del numero di esempi, oltre al prompt *multilabel few-shot 2* per verificarne la stabilità in quanto performante esattamente come l'11.

La stima della deviazione standard calcolata per ogni prompt mostra performance sovrapponibili e in linea con quanto osservato nella fase di preselezione.

Prompt	Media F1 medio	Media F1 ponderato	CV% F1 medio	CV% F1 ponderato
1	0.67 ± 0.02	0.73 ± 0.02	2.3	2.8
10	0.65 ± 0.02	0.78 ± 0.02	3.8	2.5
6	0.68 ± 0.03	0.80 ± 0.02	4.2	2.8
7	0.69 ± 0.03	0.80 ± 0.02	3.7	2.6
8	0.65 ± 0.03	0.78 ± 0.04	4.8	4.7
9	0.61 ± 0.03	0.73 ± 0.04	5.0	5.0
11	0.67 ± 0.03	0.74 ± 0.04	3.9	5.0
2	0.67 ± 0.03	0.74 ± 0.03	3.8	4.2

Tabella 5.5: KPI medi dell'ultima analisi, utilizzando Qwen3 – 4B Instruct.

Sulla base dei risultati ottenuti, il *Prompt 7* è stato scelto per le analisi successive, in quanto caratterizzato dalla combinazione più favorevole tra deviazione standard mediamente più bassa e *KPI* mediamente più elevati.

5.2. Risultati del confronto con BERT

Il confronto tra le performance del *LLM Qwen3 – 4B Instruct* e *Italian BERT XXL* è avvenuto frase per frase e sullo stesso dataset da 900 frasi. L'obiettivo di questo confronto è quello di valutare quale modello abbia le migliori prestazioni per lo stesso task di etichettatura binaria affrontato nello stesso modo. è importante considerare il fatto che *BERT* è stato sottoposto a *fine-tuning*, mentre *Qwen* era un modello non addestrato sul task specifico.

In *Figura 5.6* sono presentati i risultati della quantità di etichette a 0 o 1 nel dataset. Dato che il dataset è sbilanciato, è stato deciso di non analizzare le differenze con il parametro dell'*Accuratezza* come invece eseguito da Brianti per *BERT*.

Etichette	Classificate a 0	Classificate ad 1
Localizzazione	498	401
Tempo	660	239
Grave	591	308
Moto	845	54
Deboli	786	113
Totale	3380	1115

Tabella 5.6: Numero di elementi per etichetta, divisi per elementi classificati positivamente e negativamente nel dataset da 900 elementi.

Sotto, sono presentati i risultati. Prima è stata riportata la tabella contente i risultati di *Qwen*, *Tabella 5.7*. Successivamente è riportata la tabella con i risultati del modello *BERT*, *Tabella 5.8*.

	F1 medio	F1 ponderato
Localizzazione	0.72 ± 0.04	0.73 ± 0.03
Tempo	0.74 ± 0.01	0.80 ± 0.01
Grave	0.72 ± 0.01	0.75 ± 0.01
Moto	0.61 ± 0.06	0.89 ± 0.03
Utenti deboli	0.65 ± 0.02	0.84 ± 0.02
Media	0.70 ± 0.01	0.80 ± 0.01

Tabella 5.7: KPI per etichetta, risultati dall'analisi utilizzando LLM Qwen3 – 4B Instruct.

Etichette	F1 medio	F1 ponderato
Localizzazione	0.82	0.82
Tempo	0.89	0.91
Grave	0.74	0.77
Moto	0.67	0.88
Deboli	0.75	0.86
Media	0.77	0.85

Tabella 5.8: KPI per etichetta, risultati dall'analisi utilizzando Italian BERT XXL.

Come si evince dai risultati illustrati sopra, per il prompt utilizzato, *Qwen* risulta in performance mediamente minori su tutte le etichette a parte per Grave, Un'illustrazione più apparente si può evincere dalla *Figura 5.6*.

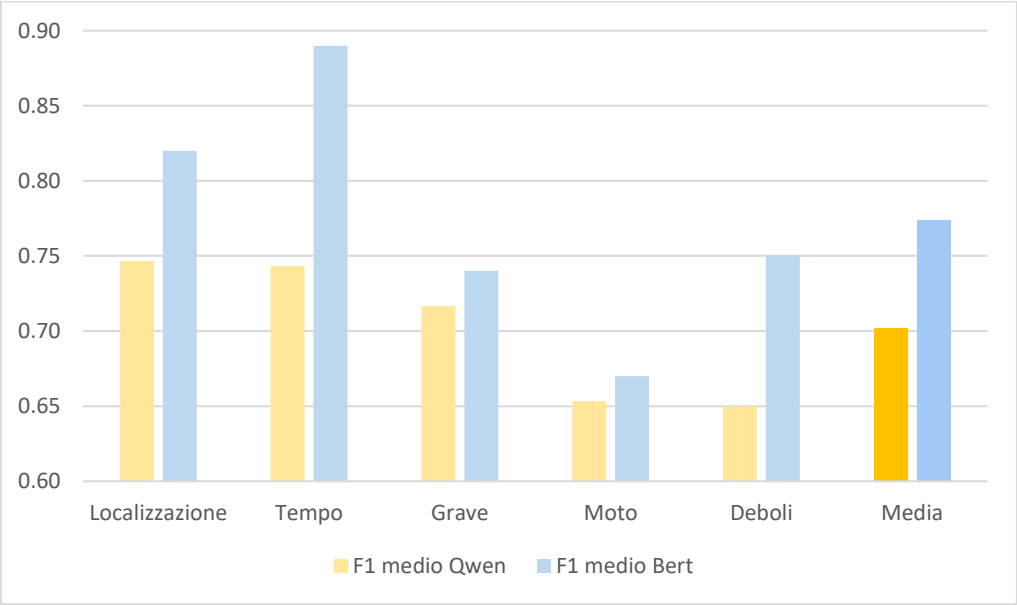

Figura 5.6: F1 score medio in comparazione tra Qwen e BERT.

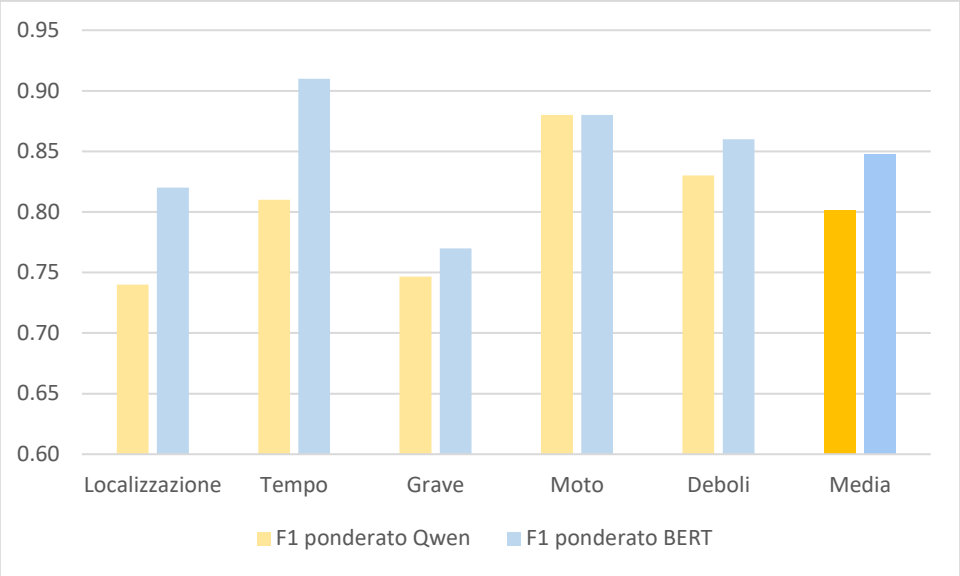

Figura 5.7: F1 score ponderato in comparazione tra Qwen e BERT.

5.3. Risultati del confronto con OpenData

Il confronto tra il database sperimentale, manipolato in modo da ottenere le informazioni target, e quello di riferimento, modificato per confrontare obiettivamente i due, è avvenuto tramite la creazione di una griglia di punti a distanza costante, attorno a cui sono state create delle aree quadrate in cui è avvenuto il calcolo dell'indice di pericolosità stradale.

I risultati ottenuti dall'utilizzo di questi due metodi di rilevazione degli incidenti stradali sono i seguenti:

I punti utilizzati escludono i punti in cui entrambi i metodi davano tutti zeri, per ragioni spiegate precedentemente durante la visualizzazione della distribuzione ottenuta, il numero di punti che compongono questo grafico sono 438.

Lo *scatter plot* calcolato è mostrato in *Figura 5.8*, oltre a questo, sono stati calcolati due valori quantitativi per descrivere la correlazione lineare tra i due metodi di misura. Il valore di *Pearson* $r = 0.4$ mostra come la correlazione lineare tra i due indici di pericolosità calcolati sia debole, mentre il *p-value* < 0.05 indica che la correlazione appena calcolata è statisticamente significativa.

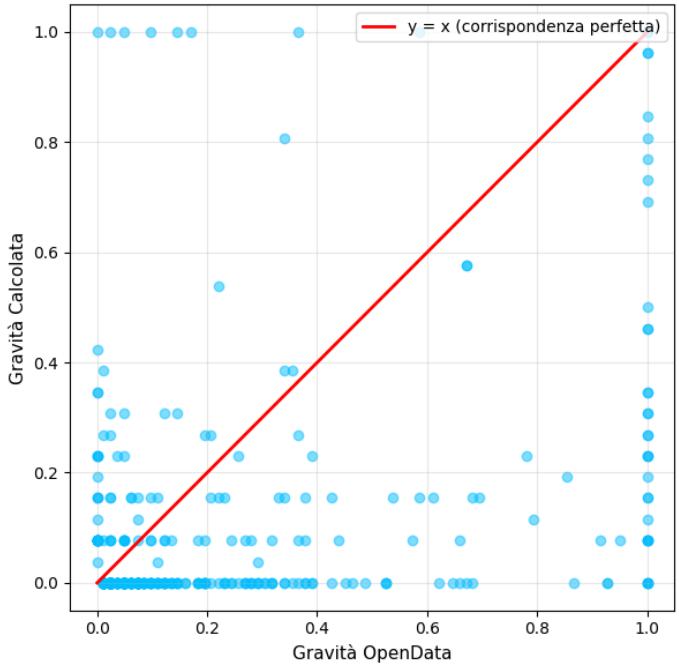

Figura 5.8: Scatter plot dell'indice di pericolosità stradale a coppie di riferimento-sperimentale.

Successivamente, è stato calcolato il *Bland-Altman plot*, uno strumento utile per confrontare due metodi di misura. Questo grafico rappresenta la differenza tra le misurazioni dei due metodi rispetto alla loro media, consentendo di valutare il livello di accordo tra di essi. Il *Bland-Altman plot* evidenzia il *bias* sistematico, che in questo caso è pari a 0.12 e i *limiti di accordo* (LOA), calcolati al 95%, questo significa che meno del 5% delle misurazioni dovrebbe ricadere all'esterno dei *LOA*. In questo caso, il numero di punti fuori dai *LOA* è 37, corrispondente all'8.4% del totale. Questo valore è superiore alla soglia attesa, indicando che l'accordo tra i due metodi non è ottimale e potrebbero non essere pienamente affidabili per tutti i dati analizzati. Il grafico è mostrato in *Figura 5.9*.

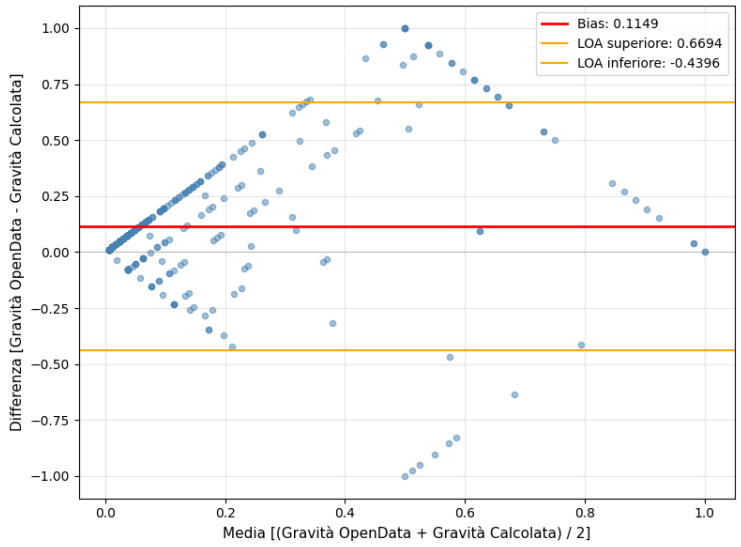

Figura 5.9: Bland-Altman plot dell'indice di pericolosità stradale a coppie di riferimento-sperimentale.

Per una visualizzazione intuitiva dell'indice di pericolosità stradale su mappa sono stati costruiti i seguenti grafici tramite Folium e i seguenti parametri:

- radius = 40: raggio di influenza di ciascun punto.
- *blur = 25*: livello di sfocatura.
- *max_zoom = 14*: livello massimo di zoom della mappa.

Figura 5.10: Heatmap dell'indice di pericolosità stradale processato da articoli di giornale.

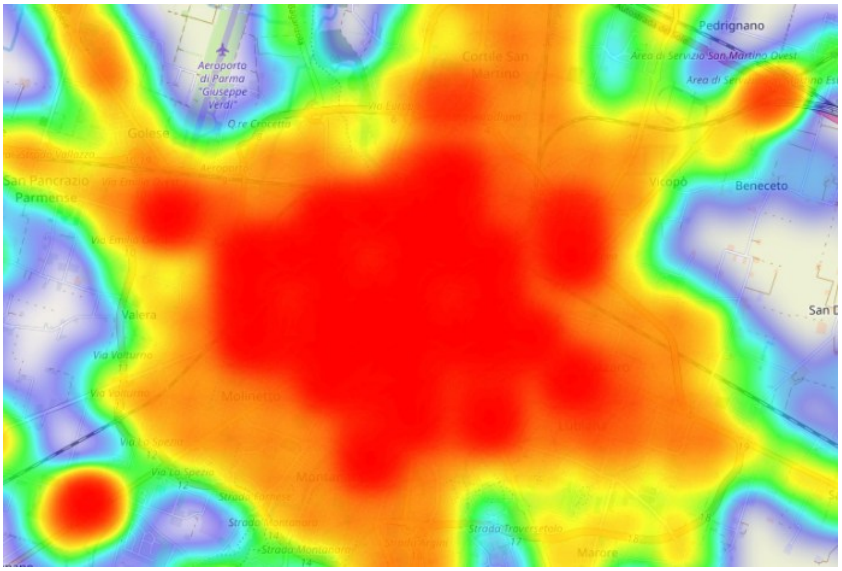

Figura 5.11: Heatmap dell'indice di pericolosità stradale processato da OpenData.

Dai due grafici, *Figura 5.10* e *Figura 5.11*, si osserva che la heatmap derivante dal dataset *OpenData* ha un livello di indice di pericolosità stradale molto più marcato rispetto alla heatmap derivante dal dataset sperimentale. Questo è può trovare una spiegazione nello sbilanciamento del numero di record tra i due dataset.

Inoltre, il grafico in *Figura 5.12* mostra la sovrastima (blu-viola) e sottostima (giallo-rosso) del metodo sperimentale rispetto al gold standard. Si osserva che, in linea con le heatmap precedenti, il metodo di misurazione sperimentale sottostima una buona parte dei punti nel comune di Parma.

Figura 5.12: Heatmap della sovrastima del metodo sperimentale (blu-viola) e sottostima (giallo-rosso) del metodo sperimentale rispetto al gold standard.

6. Conclusioni

Il lavoro svolto ha previsto due filoni principali di analisi. Sono state valutate le prestazioni di un *LLM general-purpose* di dimensioni contenute, *Qwen3 – 4B Instruct*, nella classificazione binaria di frasi riguardanti incidenti stradali, confrontandone le prestazioni con quelle del modello *Italian BERT XXL* precedentemente sottoposto a *fine-tuning*. Successivamente, è stata testata la possibilità di utilizzare articoli giornalistici come fonte alternativa agli *OpenData* per la costruzione di una mappa della pericolosità stradale.

La fase di selezione dei prompt ha mostrato come l'impiego sistematico di tecniche di *prompt engineering* possa migliorare sensibilmente la qualità dell'etichettatura binaria prodotta da un *LLM*. Tuttavia, *Qwen3 – 4B Instruct* non ha raggiunto le performance di *BERT;* lo scarto osservato è mediamente di sette punti percentuali in termini di *F1-score medio* e di cinque punti in termini di *F1-score ponderato*. Questo risultato risulta coerente con il contesto sperimentale, poiché l'etichettatura frase per frase — necessaria per il confronto con *BERT* — penalizza il *LLM*, che tende a beneficiare del contesto fornito dall'articolo completo. Si ritiene ragionevole assumere che l'utilizzo dell'intero articolo, dimostrato precedentemente dall'internato di laboratorio di Cavazzuti, o di modelli di dimensione superiore permetterebbe di ridurre tale divario, come suggerito da ulteriori osservazioni sperimentali non presentate in questo lavoro.

La seconda parte della sperimentazione ha mostrato l'intero processo di trasformazione dei dati, dall'ottenimento eseguendo lo *scraping* del quotidiano online, procedendo con etichettatura tramite *LLM*, geolocalizzazione con *NER* e *Nominatim*, infine, la manipolazione dei dati risultanti al fine di ottenere l'indice di pericolosità stradale in griglia da confrontare con i dati di riferimento. Questa sperimentazione ha permesso di valutare se i dati estratti da articoli giornalistici possano riprodurre una mappatura dell'indice di pericolosità stradale coerente con quella calcolata a partire dagli *OpenData* del Comune di Parma. La correlazione ottenuta (0.4) e l'accordo rilevato tramite *Bland-Altman* — con circa l'8.4% dei punti al di fuori dei limiti di concordanza — indicano una discreta coerenza tra le due fonti, pur in presenza di un'evidente asimmetria informativa tra dataset (rapporto 8:1 a favore del gold standard). Questa limitazione strutturale è imposta dalla natura sbilanciata dei due dataset, quindi, non aggirabile. Si ritiene importante osservare un ulteriore criticità, che riguarda il metodo di aggregazione spaziale; la mappatura su griglia creata a partire dagli articoli di giornale non riflette la granularità reale della rete stradale, in quanto gli articoli non menzionano l'indirizzo dell'incidente. Un'estensione futura potrebbe quindi adottare come unità minima la singola strada, modellando la viabilità tramite grafo ed evitando parte dell'imprecisione derivante dall'assenza di coordinate esatte negli articoli giornalistici.

Il contributo di questo studio non risiede esclusivamente nei risultati quantitativi, ma anche nella definizione del metodo; è stato mostrato come partendo dal dato grezzo non strutturato sia possibile costruire una pipeline end-to-end in grado di elaborare

automaticamente informazioni non strutturate, estraendo nuove conoscenze interpretabili e potenzialmente impiegabili per supportare decisioni e interventi mirati in tema di sicurezza stradale. L'approccio proposto costituisce un punto di partenza solido per applicazioni successive. In un primo momento, sarebbe opportuno migliorare l'accuratezza tramite l'integrazione di modelli più grandi utilizzabili anche per estrarre la localizzazione, ma anche l'impiego di altri metodi di calcolo dell'indice di pericolosità stradale. In un secondo momento, si potrebbero aumentare i dettagli come la classificazione ed evoluzione della pericolosità in base alle date temporali e l'ampliamento alla totalità del territorio nazionale.

In conclusione, i risultati ottenuti non indicano una piena equivalenza tra la metodologia proposta e quella di riferimento, ma indicano una direzione promettente di sviluppo. In futuro, gli articoli giornalistici potrebbero costituire un ulteriore strumento informativo per il tema della sicurezza stradale.

Ringraziamenti

Desidero cogliere l'occasione per esprimere alcune parole di riconoscenza verso tutti coloro che sono stati al mio fianco durante questi anni, che mi hanno sopportato e supportato, anche nei momenti difficili, che hanno scelto di criticarmi e credere in me.

Un ringraziamento speciale va a tutti gli amici e parenti che non citerò singolarmente qui sotto: vi ho pensati, ma ho deciso di non scrivere una seconda tesi su ciascuno di voi.

Vorrei ringraziare specialmente i Genitori: *Rosalba* e *Silvano*, per avermi educato, essersi presi cura di me e per aver reso possibile questo percorso.

Al mio caro amico *Mazzo*, per i tanti progetti ed episodi memorabili vissuti insieme, ma anche per tutte le uscite di rito dimenticate e alle innumerevoli corse al parco. Insomma, *Grazie al Mazzo.*

Alla *Fede*, un ringraziamento speciale per essere diventata insieme a me una singola entità secondo molte persone, ma anche per tutti i discorsi filosofici spaccacervello, per essere il vero Odisseo e per tanto altro ancora.

Alla Bene, che mi ha accompagnato e con cui sono cresciuto insieme per tutti questi anni; una persona su cui so di poter contare davvero e che ha reso dolce il mio ricordo di Malta e di molti altri luoghi.

A *Peco*, distributore di idee alternative e sovversive, grazie per avermi introdotto e fatto appassionare al mondo del teatro, e citando la cara Paola: "*Grazie, Grazie, Grazie a tutti!".* A *Dogo* e *Piero*, i più recenti, ma non troppo, compagni di avvenutre strane e tutte pazze. Infine, a tutti coloro che ho perso, ma che comunque ricordo con affetto.

7. Bibliografia

[1] A. Gov, «ACI-ISTAT: report degli incidenti stradali del 2023,» 2024. [Online]. Available: https://aci.gov.it/comunicati-stampa/aci-istat-report-degli-incidenti-stradali-2023/. [Consultato il giorno 2025].

[2] Istat, «Incidenti stradali in Italia - 2024,» 2025. [Online]. Available: https://www.istat.it/comunicato-stampa/incidenti-stradali-in-italia-2024/. [Consultato il giorno 2025].

[3] C. d. Parma, «Incidenti stradali complessivi dal 2016 al 2022,» 2022. [Online]. Available: https://opendata.comune.parma.it/dataset/incidenti-stradali-2022.

[4] E. A. Michele Tomaiuolo, *DangeRoads, apprendimento automatico per la geolocalizzazione degli incidenti stradali,* 2022-2023.

[5] M. T. Marcello Brianti, «Creazione di un modello basato su BERT per la classificazione di articoli di giornale riguardanti incidenti stradali,» 2023.

[6] T. M. G. L. Jacopo Cavazzuti, «Tirocinio modello AI,» 2025.

[7] P. d. Milano, «Atlante degli incidenti ciclistici,» 23 10 2025. [Online]. Available: https://craft.dastu.polimi.it/it/articles/15.

[8] B. Ottawa, «Interactive Maps,» [Online]. Available: https://maps.bikeottawa.ca/.

[9] J. &. C. M.-W. &. L. K. &. T. K. Devlin, «Bert: Pre-training of deep bidirectional transformers for language understanding,» 2018. [Online]. Available: https://arxiv.org/abs/1810.04805.

[10] S. Schweter, «Italian bert and electra models,» Novembre 2020. [Online]. Available: https://zenodo.org/records/4263142.

[11] A. Cloud, «Alibaba Cloud for generative AI,» [Online]. Available: https://www.alibabacloud.com/en/solutions/generative-ai?_p_lc=1.

[12] A. Cloud, «Qwen2.5-1.5B Instruct – Hugging Face,» [Online]. Available: https://huggingface.co/Qwen/Qwen2.5-1.5B-Instruct. [Consultato il giorno 2025].

[13] A. Cloud, «Qwen2.5-1.5B Instruct – Hugging Face,» [Online]. Available: https://huggingface.co/Qwen/Qwen3-4B-Instruct-2507. [Consultato il giorno 2025].

[14] B. X. Z. L. Z. Z. J. Z. Y. Z. e. a. A. Yang, «Qwen3 Technical Report, arXiv,» 2025. [Online]. Available: https://arxiv.org/abs/2505.09388.

[15] U. d. S. d. Parma, «HPC di ateneo, Servizio di Calcolo Scientifico,» [Online]. Available: https://www.hpc.unipr.it/dokuwiki/doku.php?id=calcoloscientifico:progetto.

[16] R. G. a. B. Sundheim, «Design of the MUC-6 evaluation,» in *Sixth Message Understanding Conference (MUC-6): Proceedings of a Conference Held in Columbia, Maryland, November 6-8, 1995*.

[17] BullMount, «it_nerIta_trf,» [Online]. Available: https://huggingface.co/bullmount/it_nerIta_trf/tree/main.

[18] «OpenStreetMap,» 2004. [Online]. Available: https://www.openstreetmap.org/.

[19] «Nominatim,» OpenStreetMap, [Online]. Available: https://nominatim.org/.

[20] Scikit-learn.org, «Scikit-learn,» [Online]. Available: https://scikit-learn.org/stable/getting_started.html.

[21] «Requests: HTTP for Humans™,» [Online]. Available: https://docs.python-requests.org/en/latest. [Consultato il giorno 2025].

[22] «Beautiful soup documentation,» [Online]. Available: https://www.crummy.com/software/BeautifulSoup/bs4/doc/. [Consultato il giorno 2025].

[23] «GeoPy documentation,» [Online]. Available: https://geopy.readthedocs.io/en/stable/.

[24] «Shapely documentation,» [Online]. Available: https://shapely.readthedocs.io/en/stable/.

[25] «Folium documentation,» [Online]. Available: https://python-visualization.github.io/folium/latest/.

[26] «Matplotlib: visualization with Python,» [Online]. Available: https://matplotlib.org/.

[27] A. Z. John Berryman, in *Prompt Engineering for LLMs*, O'Reilly, 2024.

[28] S. learn, «f1_score,» [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html.

[29] J. T. &. P. Jaillet, in *Introduction to Probability*, MIT OpenCourseWare, 2015.

[30] W. Navidi, in *Statistics for engineer and scientists*, McGrawHill, 2015, p. 182.

[31] dbmdz, «dbmdz/bert-base-italian-xxl-cased,» [Online]. Available: https://huggingface.co/dbmdz/bert-base-italian-xxl-cased.

[32] C. d. Parma, «Mappe interattive - Comune di Parma,» [Online]. Available: https://mappe.comune.parma.it/mokaApp/apps/STRWEB_H5/index.html. [Consultato il giorno 2025].

[33] K. S. V. M. Alexandropoulos S-AN, «Data preprocessing in predictive data mining,» *The Knowledge Engineering Review,* vol. e1, n. doi:10.1017/S026988891800036X , p. 34, 2019.

[34] G. L. Marina Sokolova, «A systematic analysis of performance measures for classification tasks,» *Information Processing & Management,* pp. 427-437, 2009.