Relazione Progetto Natural Language Processing

Francesco Stiro, 1000027019

1 Introduzione

Negli ultimi anni il Natural Language Processing ha compiuto enormi passi in avanti grazie al rilascio del paper "Attention is all you need" che portò all'avvento dei modelli di deep learning basati su architetture Transformer. In particolare, la capacità di rappresentare i testi sotto forma di embedding densi ha rivoluzionato tutti i task di NLP, come la ricerca semantica, question answering, named entity recognition e linking, e molti altri.

In particolare, i modelli only-encoder (o dual-encoder), come Sentence-BERT, sono stati progettati per rappresentare frasi e documenti in uno spazio vettoriale in cui testi semanticamente simili risultano vicini tra loro. Questo approccio consente di confrontare direttamente porzioni di testo sulla base del loro significato, ed è particolarmente efficace in compiti come il semantic search o il question answering. Questi citati sono modelli specifici, tuttavia non tutti i modelli di embedding si comportano allo stesso modo: a seconda dei dati di addestramento e delle architetture possono emergere differenze significative in termini di qualità delle rappresentazioni e velocità di esecuzione. Per valutare in modo oggettivo queste differenze, il progetto si pone diversi obiettivi:

1. Confrontare modelli di embedding: confronto tra due modelli della famiglia Sentence-BERT e il modello GTE-Large, valutandone la qualità semantica degli embedding generati.

2. Implementare un sistema di *information retrieval*: utilizzo della libreria FAISS per la ricerca dei vettori più simili a una query.

3. Valutare le performance: misurazione della bontà dei modelli sulla base delle metriche di retrieval adottate (Hit@K, MRR, Tempo di Esecuzione).

Il dataset utilizzato è SciFact (da Allen Institute for AI), reperibile al seguente link: **https://huggingface.co/datasets/allenai/scifact**

2 Dataset

SciFact è un dataset per il fact-checking scientifico, composto da claim annotati con evidenze estratte da un corpus di abstract scientifici strutturati. Il contenuto del dataset è il seguente:

1. Un insieme di claims (asserzioni) suddivise in split train, validation e test. Ogni claim è corredato da un insieme di campi:
   - `id (int32)`: una chiave numerica che rappresenta univocamente il claim.
   - `claim (string)`: il claim vero e proprio.
   - `evidence_doc_id (string)`: Il numero del documento più rilevante che supporta il claim.
   - `evidence_label (string)`: Specifica in che rapporto il documento sta con il claim. Le etichette sono due: "support" e "contrast".
   - `evidence_sentences (list of int32)`: Lista numerata delle frasi che supportano il claim.
   - `cited_doc_ids (list of int32)`: Lista degli id dei documenti citati dal claim.
2. Un corpus di oltre 5000 abstract di articoli scientifici con un solo split (quello di train). Ogni documento del corpus ha un insieme di campi:
   - `doc_id (int32)`: Una chiave numerica che rappresenta univocamente il documento.
   - `title (string)`: Il titolo del paper.
   - `abstract (list of string)`: Lista sequenziale di tutte le frasi dell'abstract. Ovviamente, l'insieme delle frasi compone l'abstract stesso.
   - `Structured (boolean)`: Indica semplicemente se il dato abstract è un abstract strutturato (cioè diviso in sezioni contrassegnate da etichette, piuttosto che un blocco di testo libero.

2.1 Preprocessing

Prima di passare alla fase di embedding e retrieval, occorre preparare il dataset in modo da semplificare la valutazione delle performance. Vengono eseguiti i seguenti step:

1. **Rimozione dei claim privi di etichetta di riferimento.** Alcuni record negli split di claims non contengono un valore valido in evidence_doc_id (stringa vuota). Li eliminiamo con un filtro automatico, così da lavorare soltanto su claim dotati di un documento di ground truth. Questo è essenziale per poter valutare le performance.

2. **Unione degli split per il retrieval.** Per il nostro task di recovery top-K non eseguiamo fine-tuning né valutazioni separate su train/validation; uniamo semplicemente i claim di train e validation in un'unica lista di query. Analogamente, utilizziamo l'intero split train del corpus per costruire l'indice FAISS sugli abstract.

Inoltre, per garantire il corretto confronto tra i documenti ritrovati da FAISS e quelli indicati come ground truth, è fondamentale mantenere l'allineamento tra gli embedding e i rispettivi doc_id. Nel nostro esperimento, abstracts e doc_ids vengono caricati nello stesso ordine dal corpus. Gli embedding degli abstract, generati in questo stesso ordine, vengono poi indicizzati con FAISS. Poiché FAISS restituisce indici numerici riferiti alla posizione degli embedding, possiamo ricavare il corrispondente doc_id semplicemente accedendo a doc_ids[i]. Questo garantisce che le metriche di valutazione siano calcolate in modo corretto.

3 Modelli utilizzati

Per questo progetto sono stati selezionati e confrontati tre modelli pre-addestrati per la generazione di sentence embeddings, ciascuno con caratteristiche diverse in termini di performance, accuratezza semantica e tempi di calcolo. L'obiettivo del confronto è valutare la loro efficacia nell'individuare il documento più rilevante rispetto a un claim, in uno scenario di information retrieval.

3.1 thenlper/gte-large

gte-large (General Text Embeddings) è un modello di tipo encoder-only rilasciato dal gruppo The NLP Group. È stato addestrato per generare rappresentazioni semantiche universali, ottimizzate per una varietà di compiti NLP tra cui semantic similarity, retrieval e clustering. Si distingue per l'elevata accuratezza, ma ha una latenza significativamente maggiore rispetto ad altri modelli più leggeri. Non fa parte della famiglia Sentence-BERT, ma è stato incluso proprio per osservare come si comporta un modello con maggiore precisione ma minore efficienza computazionale.

3.2 all-MiniLM-L6-v2

Questo modello, parte della famiglia Sentence-BERT, è progettato per essere estremamente leggero e veloce. Utilizza un'architettura MiniLM con solo 6 layer, il che lo rende particolarmente adatto per scenari real-time o su dispositivi con risorse limitate. Pur essendo meno preciso rispetto a modelli più grandi, rappresenta un ottimo compromesso tra velocità ed efficacia semantica.

3.3 all-mpnet-base-v2

Anche questo modello appartiene alla famiglia Sentence-BERT, ma è basato sull'architettura MPNet, che combina i punti di forza di BERT e di permuted language modeling. Il modello "all-mpnet-base-v2" ha dimostrato ottime performance su diversi benchmark di semantic textual similarity e sentence embeddings. Rispetto a MiniLM è più lento, ma garantisce embedding semantici più ricchi, ed è spesso considerato uno tra i modelli che performa meglio di Sentence-BERT.

La selezione di questi modelli è motivata dalla volontà di confrontare:

- Un modello di nuova generazione ad alta accuratezza (GTE),
- Un modello ottimizzato per essere veloce (MiniLM),
- Un modello bilanciato in termini di accuratezza ed efficienza (MPNet).

I risultati sperimentali possono darci un'idea sulle loro performance, e accorgerci se conviene o meno utilizzare modelli più dispendiosi (come GTE-large) piuttosto che modelli meno accurati ma più veloci, dipendentemente dal task che vogliamo affrontare.

4 Metodologia

La metodologia per il confronto tra modelli segue una pipeline modulare, composta da tre fasi principali:

1. **Generazione degli embedding**. In questa fase, ogni claim (asserzione) e ogni abstract del corpus viene trasformato in un vettore denso a bassa dimensionalità (embedding) tramite un modello pre-addestrato. I vettori prodotti catturano le relazioni contestuali tra le parole, e di conseguenza la semantica dell'intero abstract. Come sappiamo, infatti, vicinanza nello spazio vettoriale implica similarità semantica.
2. **Indicizzazione con FAISS**. Gli embedding degli abstract vengono memorizzati in una struttura indicizzata costruita con la libreria FAISS (Facebook AI Similarity Search), progettata per le ricerche di similarità tra vettori su larga scala in modo molto efficiente. In particolare, viene utilizzato un indice di tipo IndexFlatL2, che consente di calcolare la distanza euclidea tra i vettori.
3. **Retrieval e confronto con il ground truth**. Per ogni claim, viene eseguita una ricerca nel database FAISS per identificare i documenti con embedding più vicini (Top-K). Gli identificativi dei documenti restituiti vengono confrontati con il campo "evidence_doc_id" del dataset, che rappresenta il documento etichettato come corretto. Da questo confronto derivano le metriche di performance, che vedremo in seguito.

Questa pipeline, mantenuta identica per ogni modello testato, garantisce coerenza nell'esperimento e consente di confrontare i modelli in modo equo, sia in termini di accuratezza semantica che di prestazioni computazionali.

Ricollegandoci al punto 2, per l'indicizzazione degli embedding è stato utilizzato un indice IndexFlatL2, che calcola la distanza euclidea tra vettori. Sebbene la metrica più naturale per confrontare embedding semantici sia la similarità del coseno, questa non è supportata direttamente da FAISS. Il motivo per cui non la supporta è che FAISS è stato progettato principalmente per essere molto performante su grandi volumi di dati, sfruttando anche l'architettura della GPU per migliorare l'efficienza computazionale del task retrieval. Si basa su metriche che rispettano proprietà come la disuguaglianza triangolare, come la distanza L2. La similarità del coseno, d'altra parte, rompe alcuni assunti su cui FAISS basa le sue ottimizzazioni. Tuttavia, esiste un accorgimento pratico che consente di aggirare questa limitazione: normalizzando gli embedding, la distanza euclidea L2 diventa monotonicamente equivalente alla distanza del coseno. Di conseguenza, l'ordinamento dei risultati di retrieval non cambia e le metriche Hit@K e MRR non vengono influenzate. Questo nasce dalla seguenti considerazioni.

Dati due vettori u, v, il coseno dell'angolo tra i due vettori è dato da:

$$\cos(u, v) = \frac{u \cdot v}{\|u\| \cdot \|v\|}$$

Se i vettori sono normalizzati, cioè $\|u\| = 1$ e $\|v\| = 1$, si ha:

$$\cos(u, v) = u \cdot v$$

Per la distanza euclidea, invece:

$$\|u - v\|^2 = \|u\|^2 + \|v\|^2 - 2(u \cdot v)$$

Se i vettori sono normalizzati:

$$\|u - v\|^2 = 1 + 1 - 2(u \cdot v) = 2(1 - u \cdot v)$$

e poiché $u \cdot v = \cos(u, v)$:

$$\|u - v\|^2 = 2(1 - \cos(u, v))$$

Abbiamo dimostrato che le due metriche sono **monotonicamente equivalenti**. Questa trasformazione garantisce che l'indice restituisca lo stesso ordinamento che si otterrebbe utilizzando direttamente la similarità del coseno, pur sfruttando la maggiore efficienza di calcolo offerta da FAISS. Questo vale naturalmente dall'assunzione che i vettori siano normalizzati.

5 Valutazione e risultati

Per valutare l'efficacia degli embedding generati dai modelli, è stato scelto un approccio basato su metriche di retrieval. Queste metriche non misurano direttamente la "bontà" degli embedding in senso assoluto, ma ne valutano l'efficacia

nel recuperare il documento corretto dato un claim. In questo contesto, sono state utilizzate le seguenti metriche:

1. **Hit@K**
   Questa metrica misura la frazione di volte in cui il documento corretto si trova tra i primi k documenti più simili individuati dal modello.

   $$\text{Hit@K} = \frac{\#\text{claim con doc corretto nei top-K}}{\#\text{ totale dei claim}}$$

   Questa metrica è utile per valutare la capacità degli embedding di mettere in alto nel ranking i documenti semanticamente rilevanti.

2. **MRR** (Mean Reciprocal Rank)
   La metrica MRR valuta quanto in alto nel ranking viene posizionato il documento corretto.

   $$\text{MRR} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{rank_i}$$

   dove $rank_i$ è la posizione del documento corretto nella lista ordinata dei risultati per il claim i. MRR punisce di più i documenti trovati in posizione bassa, e prem,ia i match nella posizioni alte, specialmente TOP-1.

3. **Tempo di esecuzione**
   È stato inoltre misurato il tempo impiegato da ciascun modello per generare gli embedding e completare l'intero processo di retrieval. Questa metrica è utile per confrontare l'efficienza computazionale dei modelli, soprattutto in scenari dove la velocità è un requisito critico.

5.1 Risultati sperimentali

Per ciascun modello, sono stati calcolati i valori di Hit@K con ($K = 1, 3, 5, 10$) e MRR, oltre al tempo totale di esecuzione (in secondi), per confrontare sia l'efficacia semantica che l'efficienza computazionale. Di seguito i risultati in forma tabellare:

Table 1: Confronto tra modelli in termini di Hit@K, MRR.

Modello	**Hit@1**	**Hit@3**	**Hit@5**	**Hit@7**	**Hit@10**	**MRR**
`thenlper/gte-large`	0.533	0.730	0.799	0.829	0.860	0.642
`all-mpnet-base-v2`	0.499	0.658	0.733	0.774	0.802	0.595
`all-MiniLM-L6-v2`	0.453	0.649	0.725	0.765	0.802	0.567

Table 2: Confronto tra modelli in termini di tempi di esecuzione

Modello	**Tempo (s)**
`thenlper/gte-large`	36.7
`all-mpnet-base-v2`	15.1
`all-MiniLM-L6-v2`	4.5

Per una comprensione più immediata, di seguito i plot relativi alle metriche di performance.

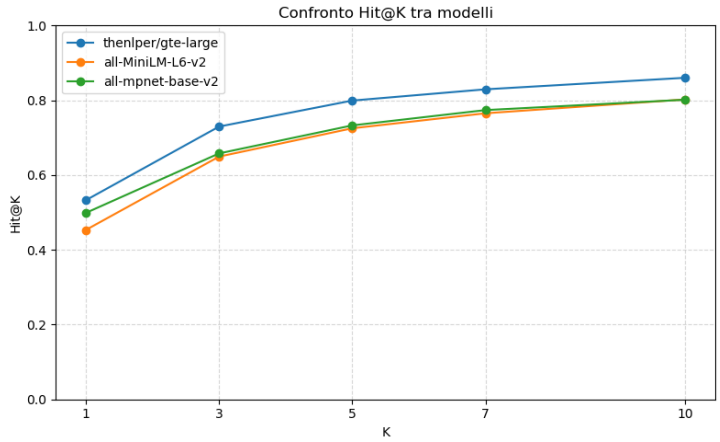

Figure 1: Hit@K Comparison

La figura 1 mostra il confronto delle performance dei modelli in termini di Hit@K. Si osserva che gte-large performa sistematicamente meglio per qualunque valore di K. Per quanto riguarda gli altri due modelli, invece, le differenze in performance tendono ad assottigliarsi all'aumentare di K. Per $K = 1$ il modello "all-mpnet-base-v2" performa meglio, cioè restituisce più volte il documento di riferimento. All'aumentare di K, entrambi performano pressochè allo stesso modo.

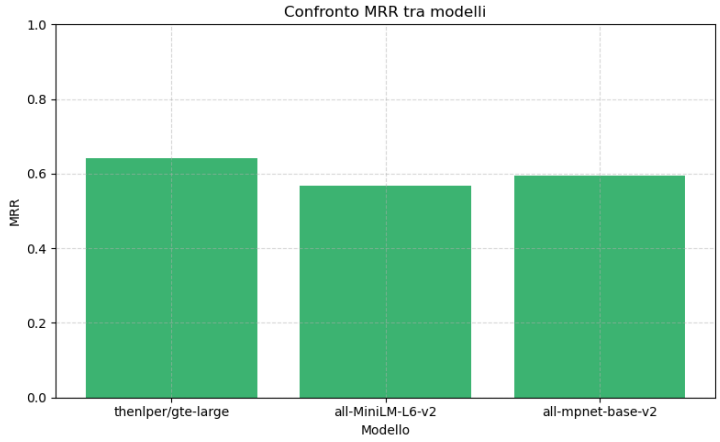

Figure 2: MRR Comparison

La figura 2 mostra il confronto tra i tre modelli in termini di MRR (Mean Reciprocal Rank), una metrica che valuta la posizione della prima risposta corretta. Anche in questo caso, "thenlper/gte-large" si conferma il modello più performante, ottenendo un MRR superiore rispetto agli altri. I modelli "all-MiniLM-L6-v2" e "all-mpnet-base-v2" mostrano risultati comparabili tra loro, leggermente a favore per il secondo.

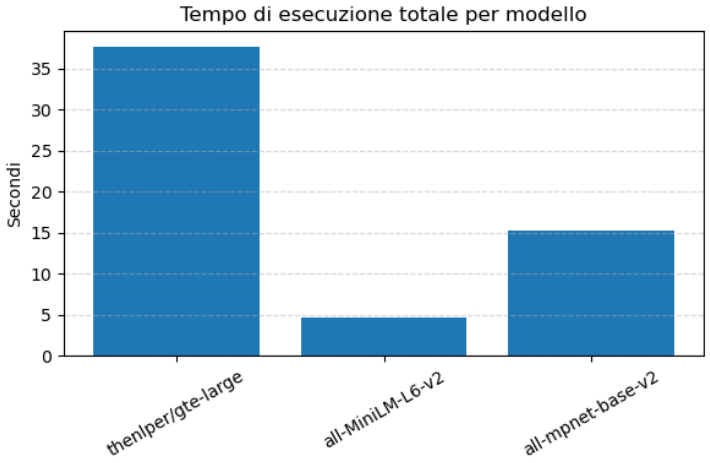

Figure 3: Execution Time Comparison

La figura 3 presenta il confronto dei tre modelli in termini di tempo di esecuzione totale. Si osserva che "thenlper/gte-large", pur essendo il più preciso secondo le metriche precedenti, è anche il più lento, con un tempo significativamente maggiore rispetto agli altri due. In particolare è più lento di circa un fattore 2 rispetto a "all-mpnet-base-v2" e di un fattore 7 rispetto a "all-MiniLM-L6-v2". Al contrario, "all-MiniLM-L6-v2" risulta il più efficiente, seguito da "all-mpnet-base-v2" (che a confronto, è piu lento di un fattore 3). Questo evidenzia un compromesso tra accuratezza e velocità di esecuzione che può influenzare la scelta del modello in contesti applicativi sensibili ai tempi di risposta.

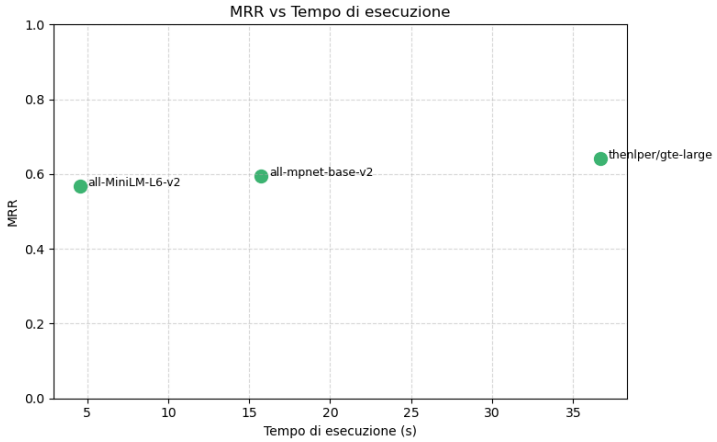

Figure 4: MRR vs Execution Time Comparison

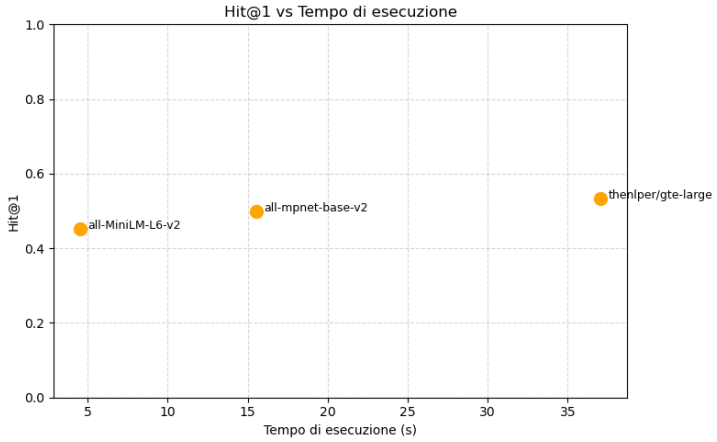

Figure 5: Hit@1 vs Execution Time Comparison

Le Figure 4 e 5 mostrano rispettivamente la relazione tra MRR e Hit@1 rispetto al tempo di esecuzione. I due grafici confermano un andamento molto simile: "thenlper/gte-large" ottiene le migliori performance in termini di accuratezza, ma a fronte di un tempo di esecuzione significativamente più elevato. Questi grafici suggeriscono che, nonostante i risultati superiori in termini di precisione, il rapporto tra accuratezza e prestazioni penalizza fortemente gte-large, il cui tempo computazionale risulta sproporzionato rispetto al guadagno ottenuto. In contesti in cui la latenza o l'efficienza sono fattori critici, modelli come all-mpnet-base-v2 o all-MiniLM-L6-v2 rappresentano soluzioni più bilanciate tra qualità e costo computazionale. Naturalmente vorremmo che il nostro

modello (rispetto a questi ultimi due plot) sia più vicino possibile al punto (0,1) in alto a sinistra. Questo perchè avrebbe un tempo di esecuzione più basso e una precisione maggiore.

6 Conclusioni

In questo progetto è stato analizzato il comportamento di diversi modelli di embedding pre-addestrati, con l'obiettivo di valutarne l'efficacia nel compito di information retrieval tra coppie claim - abstract, utilizzando il dataset SciFact. Sono stati confrontati modelli di natura diversa, tra cui "thenlper/gte-large" e "all-MiniLM-L6-v2", valutandoli sia in termini di accuratezza (mediante metriche come Hit@K e MRR) che di tempo di esecuzione. I risultati hanno mostrato che il modello "thenlper/gte-large" ottiene le migliori prestazioni in termini di accuratezza, superando i modelli della famiglia Sentence-BERT come "all-MiniLM-L6-v2" e "all-mpnet-base-v2". Tuttavia, questa maggiore precisione comporta un costo computazionale significativamente superiore. Al contrario, modelli come "all-MiniLM-L6-v2" offrono un eccellente compromesso tra accuratezza e velocità, rendendoli adatti a scenari in cui le risorse computazionali sono limitate o è richiesta bassa latenza. Sorprendentemente, quest'ultimo, riesce ad avere prestazioni molto simili ad un modello bilanciato come "all-mpnet-base-v2", essendo pure tre volte più veloce.

6.1 Sviluppi futuri

Un possibile sviluppo di questo lavoro consiste nell'integrare ulteriori modelli di embedding, eventualmente addestrati su dati scientifici. Inoltre, sarebbe interessante esplorare l'impatto della struttura del documento (campo structured) o del titolo (title) sul processo di retrieval, per potenziare ulteriormente le performance del sistema. Si potrebbe inoltre cercare di fare classificazione sul campo "evidence_label" per capire se un dato documento supporta o contrasta il claim.

7 Link utili

1. https://github.com/facebookresearch/faiss
2. https://sbert.net/
3. https://huggingface.co/thenlper/gte-large
4. https://huggingface.co/datasets/allenai/scifact