Transmedia Library: un Approccio RAG Strictly-Grounded per la Scoperta di Relazioni e l'Interrogazione della Conoscenza

Ettore Perri
Progetto Transmedia Library
Desme Digital S.R.L.
Email: ettore@desmedigital.it
Data: 16 luglio 2026 ORCID: 0009-0001-6057-655X

***Sommario*—Questo documento descrive l'architettura retrieval-augmented generation (RAG), la strategia di embedding e i meccanismi di scoperta di relazioni implementati nella Transmedia Digital Library — un sistema che indicizza item multimediali eterogenei (libri, film, registrazioni audio, risorse web) ed espone tre funzionalità basate su IA: (1) una pipeline di recupero ibrido per l'assemblaggio del contesto del chatbot, (2) un motore di scoperta di relazioni tra item, e (3) un'interfaccia conversazionale ancorata esclusivamente al materiale sorgente indicizzato. Diverse scelte progettuali non standard sono documentate e motivate, tra cui uno schema documentale a due livelli, la pesatura gaussiana per prossimità sui punteggi RRF delle keyword, il citation boosting graduale tramite saturazione esponenziale, la fusione di similarità asimmetrica tramite aggregazione MAX, e la selezione del modello scope-aware che accoppia bersaglio di recupero e livello di accesso in un'unica superficie API.**

***Index Terms*—Retrieval-Augmented Generation, Ricerca Vettoriale, Information Retrieval, Recupero Ibrido, Reciprocal Rank Fusion, Grafi di Conoscenza, Scoperta di Relazioni, Embeddings, Modelli Linguistici di Grandi Dimensioni, IA Conversazionale.**

I. Panoramica dell'Architettura del Sistema

Il sistema è costruito su un'applicazione Flask supportata da un database relazionale e da un vector store ChromaDB (istanza cloud). Tutta l'inferenza IA è delegata a un provider esterno di large language model tramite l'Anthropic SDK. Il vector store utilizza la distanza coseno con indicizzazione HNSW [1], e gli embedding testuali sono prodotti da un modello sentence-transformer [2], [3]. Il pattern architetturale che combina recupero denso e modello generativo segue il framework retrieval-augmented generation (RAG) [4].

Ciascun item multimediale può avere una o più *risorse*: file caricati (PDF, EPUB, audio, video, documenti Office) o risorse URL esterne. Prima che un item diventi ricercabile e conversabile, deve essere *analizzato*: il testo viene estratto da ciascuna risorsa, viene generato un riassunto per-risorsa tramite IA, e tutto il materiale viene indicizzato in ChromaDB. Un riassunto finale a livello di item viene quindi generato a partire dalla concatenazione dei riassunti per-risorsa.

II. Schema dei Documenti: Indicizzazione a Due Livelli

Decisione progettuale chiave: i documenti sono indicizzati a due granularità distinte al servizio di task di recupero differenti.

A. Documenti Riassunto (`doc_type = 'summary'`)

Un documento riassunto per ciascuna risorsa (upload o URL) più un documento di metadati a livello di item. I documenti riassunto contengono:

- Il riassunto condensato generato dall'IA della risorsa (fino a 4.000 caratteri).
- Per i documenti a livello di item: titolo + descrizione + riassunto IA complessivo.

I documenti riassunto servono la *ricerca di similarità tra item*. Sono rappresentazioni semanticamente dense e ripulite dal rumore. Poiché sono generati da un large language model condizionato sul testo completo, catturano l'essenza concettuale piuttosto che il vocabolario incidentale.

B. Documenti Chunk (`doc_type = 'chunk'`)

Testo grezzo estratto suddiviso in finestre sovrapposte:

```
CHUNK_SIZE    = 1500 caratteri
CHUNK_OVERLAP =  150 caratteri
```

I chunk sono memorizzati in *minuscolo* per abilitare la corrispondenza case-insensitive tramite il filtro `$contains` di ChromaDB. Il testo nel case originale è conservato in un campo di metadati `original_text` e usato per la visualizzazione.

I documenti chunk servono il *recupero del contesto per il chatbot*. Conservano i dettagli verbatim che i riassunti possono omettere: date specifiche, nomi, citazioni, misurazioni.

C. Razionale della Separazione

L'uso dei riassunti per la ricerca di similarità evita un problema noto: i documenti lunghi contengono molti argomenti, quindi i loro embedding a livello di chunk sono rumorosi e il loro centroide è attirato verso contenuti incidentali ad alta frequenza. Un embedding di riassunto focalizzato è un proxy più fedele dell'identità concettuale dell'item. Al contrario, l'uso dei chunk grezzi per il recupero del chatbot garantisce che il modello riceva evidenza verbatim anziché uno strato intermedio parafrasato.

III. Pipeline di Recupero Ibrido

Quando un utente invia una query al chatbot, il sistema recupera fino a 30 chunk tramite una *pipeline ibrida* che combina ricerca vettoriale densa e ricerca esatta per keyword, fuse tramite Reciprocal Rank Fusion (RRF) [5]. La pipeline attinge ai fondamenti classici dell'information retrieval [6] e alle recenti architetture di recupero aumentate da LLM [4], [7], [8].

A. Passaggio Denso

Una query di similarità vettoriale è eseguita su tutti i chunk appartenenti all'item bersaglio:

```
n_dense = max(n_results * 3, 12)   # fattore di sovra-reperimento 3x
col.query(query_texts=[query_text], n_results=n_dense, where=chunk_where)
```

La distanza coseno viene convertita in similarità: `score = 1.0 - distance`. I risultati sono memorizzati in `dense_hits`.

B. Passaggio per Keyword con Estrazione RAKE

Contributo originale: estrazione di keyword tramite RAKE con pesatura gaussiana per prossimità sui punteggi RRF.

Le keyword sono estratte dalla query tramite RAKE (Rapid Automatic Keyword Extraction) [9] con liste di stopword in italiano e inglese. Solo la frase RAKE più in alto in classifica viene utilizzata; i suoi singoli token diventano l'insieme di keyword.

```
Query: "Cosa c'entrano i Promessi Sposi"
Frase RAKE top: "promessi sposi"
Keywords: ["promessi", "sposi"]
```

Un filtro `$contains` recupera tutti i chunk dell'item che contengono *tutte* le keyword estratte. Per una singola keyword si usa un filtro `$contains` diretto; per più keyword si applica una congiunzione `$and`.

Questo garantisce che le corrispondenze verbatim non vengano *mai perse* anche quando la loro distanza di embedding è elevata — un fallimento comune per entità nominate, nomi propri rari e terminologia specifica di dominio.

C. Pesatura Gaussiana per Prossimità

Contributo originale: pesatura gaussiana dei punteggi RRF delle keyword basata sulla distanza di co-locazione.

Un chunk che contiene tutte le keyword della query è un segnale più forte se tali keyword appaiono *vicine* nel testo. Un chunk in cui "promessi" appare nella prima frase e "sposi" nell'ultima è una corrispondenza contestuale più debole rispetto a uno in cui entrambi compaiono nella stessa proposizione.

Due funzioni implementano ciò. `_calculate_gap(text, keywords)` restituisce la distanza in caratteri tra le posizioni della prima e dell'ultima keyword in `text`, restituendo ∞ se qualche keyword è assente:

```
gap = last_keyword_position - first_keyword_position
```

`_proximity_factor(gap_prompt, gap_chunk, sigma=20.0)` applica una pesatura gaussiana che confronta la dispersione delle keyword nella query con quella nel chunk:

$$\text{proximity\_factor} = \exp\left(-\frac{(gap_{chunk} - gap_{prompt})^2}{2\sigma^2}\right) \tag{1}$$

Con $\sigma = 20$ caratteri:

- $gap_{diff} = 0$ (la dispersione nel chunk corrisponde esattamente a quella nella query): fattore $= 1.000$.
- $gap_{diff} = 20$ caratteri: fattore ≈ 0.607.
- $gap_{diff} = 40$ caratteri: fattore ≈ 0.135.
- $gap_{diff} = 60$ caratteri: fattore ≈ 0.011.

Il proximity factor è applicato come *moltiplicatore sul punteggio RRF delle keyword* prima della fusione, non come reranker post-hoc. Ciò preserva la stabilità di rango di RRF introducendo al contempo un segnale di qualità continuo per le corrispondenze keyword.

Il razionale del confronto rispetto a gap_{prompt} anziché zero è che la query stessa può avere keyword disperse (ad esempio "battaglia di Lepanto 1571" ha keyword a 20 caratteri di distanza). Un chunk che replica quella dispersione è ugualmente compatto e non deve essere penalizzato.

D. Fusione RRF

Reciprocal Rank Fusion combina le due liste ordinate:

$$\text{rrf}(rank, k = 60) = \frac{1}{k + rank + 1} \tag{2}$$

Per ciascun documento che appare in una delle due liste:

```
rrf = rrf_dense + rrf_keyword * proximity_factor
```

La costante $k = 60$ è il parametro standard di RRF, che fornisce stabilità rispetto alle fluttuazioni di rango in cima alla lista.

E. Prenotazione di Slot per Keyword

Decisione progettuale: rappresentazione garantita per i chunk corrispondenti alle keyword.

Dopo la fusione, l'output finale è assemblato con un meccanismo di *prenotazione di slot*:

```
kw_slots = min(len(kw_entries), max(1, n_results // 2))
final = kw_entries[:kw_slots] + dense_entries[:n_results - kw_slots]
```

Al massimo la metà degli slot di output sono riservati ai chunk corrispondenti alle keyword. Gli slot rimanenti sono riempiti dai chunk solo-densi con il punteggio più alto. Ciò previene un fallimento comune in cui una query contenente un'entità rara non produce corrispondenze esatte nei top-N semplicemente perché l'embedding dell'entità è sottorappresentato nella distribuzione di addestramento.

IV. Architettura del Chatbot

A. Assemblaggio del Contesto

Il system prompt è assemblato per-turno a partire da quattro strati:

1) *Record item*: titolo, media type, descrizione, riassunto IA a livello di item.
2) *Riassunti per-sorgente*: il riassunto generato dall'IA di ciascuna risorsa analizzata, etichettato con un source ID stabile (`resource_N`).
3) *Passaggi rilevanti*: fino a 30 chunk recuperati dalla pipeline ibrida per la query corrente, etichettati con il loro document ID ChromaDB (`upload_N_chunk_M` o `url_N_chunk_M`).
4) *Passaggi dei turni precedenti*: chunk citati nei turni precedenti della conversazione ma non nel set di recupero corrente, per garantire la continuità delle citazioni in un dialogo multi-turno.

Ciascuna sorgente in ogni strato riceve un *identificatore stabile* che persiste per tutti i turni di una conversazione. Il modello è istruito a citare ogni affermazione fattuale usando uno shortcode inline:

```
[fonte id="upload_3_chunk_0"
    document="Biografia.pdf"]
```

B. Selezione del Modello Scope-Aware

> **Decisione progettuale:** l'endpoint di chat espone un modello logico differente per ciascuno scope, codificando sia il bersaglio di recupero sia il livello di accesso nel nome del modello.

Invece di esporre un singolo LLM configurabile, l'API pubblica di chat espone un catalogo di modelli *scope*. Il parametro `model` in ciascuna richiesta di chat determina sia il bersaglio di recupero sia il livello di autorizzazione:

- `item_<id>_model`: singolo item multimediale. Disponibile a qualsiasi utente autenticato nella stessa organizzazione dell'item. Il contesto di chat è limitato alle risorse dell'item stesso e ai chunk restituiti da `get_combined_item_context` (fino a 20 chunk primari più fino a 7 passaggi di connessione da ciascuno dei 5 item correlati più forti).
- `org_<id>_model`: tutti gli item analizzati in una data organizzazione. Disponibile solo agli utenti con ruolo `admin` o `superadmin`, e solo all'interno della stessa organizzazione a meno che il chiamante non sia superadmin. Il contesto è riempito da `get_org_context` (fino a 25 chunk) e il system prompt è arricchito con un header `<org_context>` che nomina l'organizzazione e istruisce il modello a usare i passaggi di metadati degli item (doc-id `item_*`) quando l'utente chiede cosa contiene il catalogo.
- `full_model`: ogni item pubblicato attraverso tutte le organizzazioni. Disponibile solo al ruolo `superadmin`. Il contesto è riempito da `get_global_context` (fino a 20 chunk); nessun header di organizzazione viene preposto.

La funzione `_resolve_scope` nel livello API esegue la risoluzione: il nome del modello viene confrontato con i pattern `^item_(\d+)_model$` e `^org_(\d+)_model$`; pattern sconosciuti restituiscono 404, scope validi da ruoli non autorizzati restituiscono 403, e risoluzioni valide producono una tupla `(scope, target)` che l'handler della richiesta usa per dispatchare il context builder appropriato.

Questo schema di denominazione ha tre vantaggi pratici rispetto a una lista piatta di modelli. Primo, rende lo *scope* dei dati visibile nella superficie API: un client non può accidentalmente interrogare un modello item e ricevere contesto a livello di organizzazione, perché il nome del modello e lo scope risolto sono accoppiati. Secondo, il controllo di accesso è locale al resolver: un singolo 403 su uno scope non autorizzato impedisce alla richiesta di raggiungere l'LLM, evitando sia inferenza sprecata sia il rischio di leak parziale tramite messaggi di errore che altrimenti menzionerebbero l'esistenza di risorse fuori scope. Terzo, poiché ciascuno scope è una superficie conversazionale a sé stante con un contratto di citazione e continuità condiviso, un client può comporli in un utilizzo multi-agente dispatchando un subagente per item che invochi il modello legato a quell'item; la chiamata org risponde nel proprio contesto e la risposta contiene incidentalmente i riferimenti agli item che i subagenti utilizzano.

C. Policy di Ancoraggio Rigoroso

Il system prompt impone un ancoraggio rigoroso:

- Il modello è esplicitamente proibito dall'usare conoscenza derivante dai dati di addestramento.
- Ogni affermazione fattuale deve portare una citazione inline.
- Le affermazioni senza citazione sono dichiarate invalide dal prompt.
- Il fallback quando l'informazione è assente: un breve riconoscimento che l'informazione richiesta non è contenuta nelle sorgenti fornite.

Ciò trasforma l'LLM da ragionatore generativo a *interfaccia di recupero vincolata* sul corpus indicizzato. Il pattern architetturale si allinea con i recenti design orientati alla trasparenza che integrano LLM con recupero e ragionamento formale [7], [8], [10].

D. Rilevamento della Lingua

La lingua dell'interfaccia utente (italiano o inglese), impostata dall'utente tramite lo stato di sessione, è iniettata nel system prompt come istruzione esplicita:

```
IMPORTANTE: Rispondi sempre in italiano. Non
    usare mai altre lingue
indipendentemente dalla lingua dei documenti
    sorgente.
```

Ciò sovrascrive la tendenza del modello a uniformarsi alla lingua dei documenti sorgente, che può differire dalla lingua dell'interfaccia utente.

E. Budget di Contesto

Con una context window di 204.096 token, l'allocazione del budget per turno è riportata in Tabella I.

Tabella I
ALLOCAZIONE DEL BUDGET DI CONTESTO PER TURNO.

Componente	Token stimati
Riserva di output	4.096
System prompt base (istruzioni)	∼350
Record item + riassunti sorgenti	∼800
Storia della conversazione (10 turni)	∼6.500
Disponibile per i chunk	∼192.350
30 chunk @ ∼300 token ciascuno	∼9.000
Utilizzo	∼4.7%

Il basso utilizzo è intenzionale: lascia un margine per item con materiale sorgente denso e conversazioni lunghe senza richiedere gestione dinamica del budget.

V. SCOPERTA DI RELAZIONI TRA ITEM

A. Panoramica

Il motore di scoperta di relazioni affronta una domanda distinta dal recupero: *dato che due item nel catalogo sono simili, qual è la natura della loro relazione, e tale relazione dovrebbe essere resa visibile agli utenti?* Il motore implementa una pipeline a quattro stadi che viene eseguita ogni volta che un item viene (ri-)analizzato:

1) **Similarità** (§V.B–V.F): punteggi di similarità pairwise vengono calcolati tra l'item sorgente e ogni altro item analizzato nell'organizzazione, usando una fusione RRF a due passaggi con boost di copertura e citazione e aggregazione MAX tra le direzioni.
2) **Acquisizione del vocabolario** (§V.G): l'insieme delle etichette precedentemente usate dai curatori umani nell'organizzazione viene estratto dalle relazioni memorizzate. Questo insieme funziona da vocabolario per-organizzazione, aggiornato continuamente man mano che i curatori confermano o rifiutano i suggerimenti.
3) **Generazione di etichette** (§V.H): un language model assegna un piccolo insieme di etichette concise e simmetriche a ciascuna coppia ad alta similarità, condizionato sul vocabolario acquisito.
4) **Conferma** (§V.I): le etichette sono salvate con un flag di conferma; questo fa sì che la relazione appaia nel pannello dei suggerimenti dell'item e nel grafo globale dell'organizzazione.

La proprietà progettuale cruciale è che gli stadi due e tre formano un ciclo chiuso: ogni relazione curata dall'umano diventa materiale di addestramento per il successivo passaggio di etichettatura, e il vocabolario non è mai congelato. Il pattern di usare similarità semantica su artefatti generati da LLM per la scoperta di relazioni è stato esplorato nella letteratura più ampia sui grafi di conoscenza [11]; il contributo qui è l'accoppiamento di quella scoperta con un loop di etichettatura condizionato dal vocabolario e guidato dall'input umano.

B. Similarità a Due Passaggi

Per ciascun item sorgente, la similarità con tutti gli altri item viene calcolata tramite due passaggi indipendenti, fusi con RRF. Nel *summary pass*, i documenti riassunto dell'item sorgente sono usati come vettori di query contro tutti i documenti riassunto dell'organizzazione, catturando similarità concettuale di alto livello. Nel *chunk pass*, i documenti riassunto dell'item sorgente sono interrogati contro tutti i documenti chunk, catturando dettagli condivisi specifici che il riassunto potrebbe non rappresentare.

Testi di query multipli (uno per ciascun documento sorgente) sono emessi simultaneamente. Per ciascun item bersaglio, viene mantenuto il miglior punteggio coseno tra tutte le righe di query.

C. Boost di Copertura

Contributo originale: moltiplicatore di copertura normalizzato per sorgente.

Gli item che sono matchati da una frazione maggiore dei documenti dell'item sorgente ricevono un *coverage boost*:

$$\text{coverage\_boost} = 1.0 + 0.15 \cdot \log(1 + fraction) \quad (3)$$

dove $fraction = hit\_count/total\_source\_queries$ è normalizzato in $[0, 1]$. L'uso della frazione (anziché del conteggio grezzo) garantisce che le sorgenti con molte risorse non dominino sulle sorgenti con poche risorse. Una trasformazione logaritmica fornisce rendimenti decrescenti: il primo documento corrispondente contribuisce più dei successivi.

D. Citation Boost Graduale

Contributo originale: citation boost a saturazione esponenziale basato sulla frequenza di menzione.

Un segnale distinto dalla similarità semantica è la *citazione esplicita*: il testo dell'item bersaglio menziona esplicitamente il titolo dell'item sorgente? Il rilevamento delle citazioni usa il filtro `$contains` di ChromaDB:

```
col.get(
    where_document={'$contains': source_title},
    where={'org_id': org_id, 'media_item_id':
        {'$ne': source_item_id}}
)
```

Piuttosto che un boost binario, il sistema conta *quanti chunk distinti* dell'item bersaglio contengono il titolo della sorgente. Un bersaglio che cita la sorgente in cinque passaggi distinti è una relazione più forte di uno che la cita una sola volta:

$$\text{citation\_boost} = 1.0 + 0.4 \cdot (1 - e^{-0.8 \cdot n_{citations}}) \quad (4)$$

I valori del boost per numero di citazioni sono riportati in Tabella II. La saturazione esponenziale è stata scelta rispetto a una scala lineare perché il valore evidenziale marginale della n-esima citazione diminuisce rapidamente: dieci citazioni in un documento lungo non sono dieci volte più significative di una citazione.

Tabella II
VALORI DEL CITATION BOOST PER NUMERO DI CITAZIONI.

Citazioni (n)	Moltiplicatore di boost
0	1.000 ×
1	1.220 ×
2	1.320 ×
3	1.360 ×
4	1.383 ×
∞	$\to$ 1.400 ×

E. Composizione del Punteggio Finale

Il punteggio RRF con boost usato per il ranking è:

```
rrf_boosted = rrf * coverage_boost *
    citation_boost
```

Il display score (mostrato come percentuale nell'interfaccia) è calcolato separatamente dalla similarità coseno grezza, con un bonus di citazione proporzionale:

```
citation_score_bonus = 0.15 * (1 - exp(-0.8 *
    n_citations))
display_score = min(best_cosine +
    citation_score_bonus, 1.0)
```

F. Similarità Asimmetrica e Aggregazione MAX

Decisione progettuale chiave: la similarità direzionale è preservata e fusa tramite MAX.

La similarità da A a B non è, in generale, uguale alla similarità da B ad A. Si consideri:

- **A** = "Francesco Hayez" (un item pittore con descrizione breve).
- **B** = "Risorgimento" (un item tematico ricco con molti documenti).

$A \to B$ può essere alta (i documenti di Hayez sono semanticamente vicini al materiale sul Risorgimento), mentre $B \to A$ può essere bassa (tra i molti documenti del Risorgimento, solo alcuni corrispondono specificamente alla breve descrizione di Hayez).

Il sistema calcola entrambe le direzioni indipendentemente (il `rag_suggestions_json` di ciascun item memorizza i suoi punteggi in uscita). Per la visualizzazione a grafo, dove gli archi sono non direzionati, i due punteggi sono fusi tramite *MAX*:

$$edge\_score = \max(score_{A \to B}, score_{B \to A}) \tag{5}$$

MAX è preferito alla media perché l'obiettivo è la *scoperta di relazioni*: se c'è evidenza forte in almeno una direzione, la relazione è significativa. La media sottovaluterebbe sistematicamente connessioni asimmetriche ma reali. Per il pannello dei suggerimenti per-item, viene mostrato il punteggio direzionale ($A \to B$). Ciò risponde alla domanda "quanto è rilevante questo item suggerito rispetto al contenuto dell'item corrente?" — una domanda direzionale, non simmetrica.

G. Acquisizione del Vocabolario dalle Relazioni Curate

Principio progettuale: le relazioni curate dall'umano sono il corpus di addestramento. Il modello di etichettatura è condizionato su di esse al momento dell'inferenza, quindi le scelte del curatore si propagano a ogni futuro suggerimento.

Quando un curatore conferma una relazione tra due item, le etichette che assegna sono memorizzate in una tabella dedicata `relation_label`. Al momento dell'etichettatura, il sistema interroga questa tabella per estrarre l'insieme completo di etichette attualmente in uso nell'organizzazione:

```
SELECT DISTINCT rl.label
FROM   relation_label rl
JOIN   media_item_relation r ON r.id =
    rl.relation_id
WHERE  r.is_rejected = false;
```

Il risultato è un vocabolario per-organizzazione — un elenco non ordinato e deduplicato di frasi brevi come *stessa epoca*, *personaggio condiviso*, *influenza reciproca*, *stesso autore*. Questo vocabolario viene quindi iniettato verbatim nel prompt di etichettatura come blocco `<available_labels>`, e il modello è istruito a preferirlo ogni volta che un'etichetta esistente si adatta alla relazione che sta descrivendo.

Il meccanismo è funzionalmente una forma di addestramento in-context. Non ci sono aggiornamenti di gradienti né modifiche ai parametri: il language model sottostante è lo stesso. Ciò che cambia è il *segnale di condizionamento* che il modello vede per una data organizzazione. Ciascuna azione del curatore — confermare, rifiutare, aggiungere una nuova etichetta, rimuoverne una obsoleta — cambia immediatamente il vocabolario che la successiva chiamata di etichettatura utilizzerà. Il sistema non apprende un modello generalizzabile di classificazione delle relazioni; apprende il vocabolario editoriale specifico di un'organizzazione, ancorato agli item di cui il curatore si è effettivamente occupato. Poiché il vocabolario è acquisito piuttosto che ingegnerizzato, è anche auto-stabilizzante: le etichette che nessun curatore usa semplicemente non compaiono mai nei suggerimenti, e le etichette sovrarappresentate sono rinforzate attraverso l'esposizione ripetuta. Non c'è una fase separata di "addestramento"; il workflow del curatore è l'addestramento.

H. Generazione di Etichette Simmetriche

Contributo originale: il prompt di etichettatura impone un vincolo di etichetta simmetrica, cosicché ogni etichetta generata sia valida in entrambe le direzioni della relazione.

Una singola relazione nel catalogo è memorizzata come coppia non direzionale: la riga in `media_item_relation` ha un `source_item_id` e un `target_item_id`, ma gli archi mostrati nel grafo e i suggerimenti mostrati nel pannello sono bidirezionali. Un'etichetta che abbia senso solo in una direzione — *"ispirato a"*, *"adattamento di"*, *"segue cronologicamente"* — sarebbe fuorviante nel momento in cui un utente naviga dal bersaglio alla sorgente. Il prompt di etichettatura impone pertanto un vincolo simmetrico stretto:

- Ciascuna etichetta candidata deve leggere naturalmente in entrambe le direzioni: se *"A è X di B"* è vero, allora *"B è X di A"* deve essere anch'esso vero (o almeno non contraddirlo).
- Le formulazioni direzionali sono esplicitamente vietate (*"ispirato a"*, *"adattamento di"*); si suggeriscono alternative simmetriche (*"stessa epoca"*, *"adattamento reciproco"*, *"influenza reciproca"*, *"stesso autore"*, *"personaggi condivisi"*).
- Sono richieste da due a quattro etichette per ciascuna coppia, cosicché il modello sia costretto a far emergere aspetti relazionali differenti — narrativo, tematico, formale, cronologico, personaggi condivisi — anziché collassare tutto in un'unica frase.
- La coerenza interna dell'insieme di etichette è verificata: il prompt vieta esplicitamente combinazioni che si contraddicono tra loro (ad es. *"epoca precedente"* insieme a *"artista coevo"*).
- Le etichette sono brevi (2–5 parole) e nella stessa lingua dei titoli degli item, cosicché si inseriscano naturalmente nel grafo e nel pannello dei suggerimenti.

Al modello viene chiesto di restituire un array JSON di stringhe, senza prosa e senza delimitatori markdown. Una piccola denylist di frasi generiche (ad es. *"nessuna relazione"*, *"non correlato"*) è applicata post hoc, cosicché il modello non possa sottrarsi restituendo un'etichetta vacua. Ciascuna etichetta restituita dal modello è confrontata con il vocabolario acquisito, e qualsiasi etichetta non già presente è aggiunta come candidata; la conferma del curatore promuove quindi la candidata a etichetta di vocabolario o la rimuove dal ciclo.

I. Auto-Conferma in Bulk con Persistenza Live

Contributo originale: una singola chiamata LLM processa ogni suggerimento pendente per un item e memorizza i risultati in una transazione di database orientata allo streaming.

Per un item con N suggerimenti RAG pendenti, un'implementazione ingenua emetterebbe N chiamate di etichettatura separate. Il sistema invece racchiude il lavoro in un'unica chiamata: l'item sorgente e tutti i suoi bersagli pendenti sono passati insieme, il modello restituisce un array JSON di oggetti `{item_id, labels}`, e il server itera sulla risposta per memorizzare ciascuna relazione. Ciò collassa la latenza di un tipico batch di 10–20 suggerimenti da minuti a un singolo round trip.

La chiamata in batch è di lunga durata (30–60 s per batch grandi), il che espone il sistema al timeout di connessione inattiva di MySQL. L'handler segue lo stesso pattern difensivo della pipeline di analisi (§VI.C): snapshot degli identificatori dei bersagli e dell'id utente corrente in dizionari semplici, chiusura e rimozione della sessione SQLAlchemy, esecuzione della chiamata LLM su una connessione fresca, e riapertura di una nuova sessione per le scritture. La variante streaming dello stesso endpoint, esposta a `/item/<id>/rag-auto-confirm-stream`, emette un evento `{type: 'mention', title, media_type, labels?}` per ciascun bersaglio mentre la risposta viene analizzata, cosicché l'interfaccia possa renderizzare incrementalmente ciascun suggerimento etichettato anziché attendere la risposta completa.

Un filtro di denylist post hoc (`_no_relation_phrases`) rimuove qualsiasi etichetta che neghi esplicitamente una connessione, cosicché lo step di persistenza scriva solo relazioni confermate. Ciascuna relazione persistita è o una nuova riga in `media_item_relation` (con `confirmed_by` e `confirmed_at` impostati) o un aggiornamento di una riga esistente (rimozione del rifiuto, aggiornamento delle etichette). Il risultato è un grafo delle relazioni condizionato dal vocabolario, etichettato simmetricamente e attribuito al curatore, che la successiva chiamata di etichettatura rileggerà.

VI. Pipeline di Estrazione del Testo e Sintesi

A. Estrazione

Il testo è estratto dai file caricati usando estrattori specifici per formato:

- **PDF**: `PyMuPDF` (importato come `fitz`).
- **EPUB**: `ebooklib` + `html.parser` built-in di Python.
- **Documenti Office** (DOCX, XLSX, PPTX): `python-docx`, `openpyxl`, `python-pptx`.
- **URL YouTube**: API `Supadata` con `youtube-transcript-api` come fallback (trascrizione basata su sottotitoli).
- **URL web**: `trafilatura` (rimozione del boilerplate ed estrazione del contenuto principale).
- **Immagini**: LLM con capacità visiva con un prompt di trascrizione-o-descrizione.

Al momento il sistema non implementa speech-to-text general-purpose per i file audio o video caricati; tali caricamenti sono accettati a livello di storage ma il loro contenuto testuale non viene estratto.

Il testo estratto è memorizzato nella cache in file sidecar `.ai.metadata` (JSON) adiacenti ai file caricati. Le successive ri-analisi riutilizzano l'estrazione cached a meno che `force_recompute=True` non sia impostato.

B. Sintesi

Ciascuna risorsa è riassunta indipendentemente con un input massimo di 120.000 caratteri (∼30.000 token). I riassunti per-risorsa sono quindi concatenati per produrre un riassunto finale a livello di item. Questo approccio gerarchico garantisce che il riassunto finale rifletta tutte le sorgenti proporzionalmente, anziché essere dominato dal documento più lungo. Lo stesso principio architetturale — comporre artefatti generati da LLM a granularità multiple — è adottato in framework recenti per combinare modelli generativi con risolutori simbolici [8] e per estrarre programmi logici dagli output di LLM [10].

Il prompt di sintesi istruisce il modello a produrre 3–5 frasi in testo semplice, nella lingua dell'input, catturando gli argomenti principali e lo scopo.

C. Streaming del Progresso di Analisi

La pipeline di analisi espone il progresso tramite *Server-Sent Events (SSE)*. Man mano che ciascuna risorsa viene processata, il server trasmette eventi JSON:

```
{"type": "source", "index": 2, "total": 5,
    "label": "Wikipedia - Hayez", "step":
    "summarizing"}
{"type": "final", "step": "summarizing"}
{"type": "done", "ai_summary": "...",
    "ai_analyzed_at": "12/05/2026 17:45"}
```

Il frontend aggiorna l'interfaccia in tempo reale senza ricaricare la pagina, mostrando righe di progresso per-sorgente con stati di spinner/checkmark.

VII. Riepilogo dei Contributi Originali

La Tabella III riassume i contributi originali introdotti dall'architettura. Framework concettuali strettamente correlati per combinare LLM con recupero strutturato e ragionamento simbolico sono stati sviluppati nella letteratura recente [4], [7], [8], [10].

VIII. Note di Implementazione

- **Vector store**: ChromaDB Cloud, indice HNSW, spazio coseno.
- **Modello di embedding**: default ChromaDB (all-MiniLM-L6-v2 o equivalente, gestito da ChromaDB Cloud).
- **LLM**: MiniMax-M2.7 tramite Anthropic SDK [12].
- **Context window**: 204.096 token.
- **Estrazione keyword**: `rake-nltk` con liste di stopword in italiano e inglese.
- **Parametri dei chunk**: 1.500 caratteri, sovrapposizione di 150 caratteri ($\sim$10%).
- **Costante RRF**: $k = 60$ (standard).
- **Proximity sigma**: 20 caratteri.
- **Tasso di saturazione citation**: $k = 0.8$ (costante di decadimento esponenziale).
- **Max chunk per turno di chatbot**: 30.
- **Input massimo per sintesi**: 120.000 caratteri per risorsa.

IX. Conclusioni

La Transmedia Library implementa un approccio RAG strictly-grounded in cui diverse scelte progettuali non standard sono motivate e documentate. Lo schema documentale a due livelli isola la similarità dal question-answering, permettendo a ciascun task di recupero di essere ottimizzato indipendentemente. La pesatura gaussiana per prossimità sui punteggi RRF delle keyword premia la co-locazione compatta delle keyword rispetto alla dispersione della query stessa, un segnale poco costoso che si integra in modo pulito con la rank fusion. I boost di copertura e citazione trasformano una nozione binaria di match in un segnale di forza continuo che rispetta la normalizzazione per lunghezza della sorgente. L'aggregazione MAX preserva l'evidenza asimmetrica quando si fondono punteggi di similarità direzionale in archi non direzionali del grafo. La policy di ancoraggio rigoroso trasforma l'LLM sottostante da ragionatore generativo a interfaccia di recupero vincolata i cui output sono completamente attribuibili al materiale sorgente indicizzato. La selezione del modello scope-aware accoppia il bersaglio di recupero e il livello di accesso in un'unica superficie API, rendendo lo scope dei dati visibile ai client e centralizzando il controllo di autorizzazione prima di qualsiasi chiamata di inferenza. La pipeline di scoperta delle relazioni chiude un ciclo in cui ogni relazione curata dall'umano diventa materiale di addestramento in-context per la successiva chiamata di etichettatura: il vocabolario editoriale del curatore è l'unico segnale che il sistema ha su quali relazioni siano rilevanti, e il sistema usa quel segnale senza alcuna fase di ri-addestramento.

L'architettura è limitata da alcuni vincoli ben definiti. Primo, la pipeline di chat si rivolge a un singolo item multimediale per conversazione; le query cross-item sono attualmente affrontate dal motore di scoperta delle relazioni e dalla visualizzazione a grafo, non dal livello di chat. Secondo, i file audio e video caricati sono memorizzati ma non trascritti; solo il percorso URL di YouTube fornisce trascrizioni basate su sottotitoli. Terzo, il sistema non è stato ancora valutato contro un benchmark quantitativo di recupero: le scelte progettuali sono motivate da ragionamento analitico e da failure mode osservati, non da ablation contro relevance judgement etichettati. Il lavoro futuro affronterà tutti e tre. Un'estensione naturale è un livello di chat che opera su un set di recupero fuso tratto dagli item più simili, portando il segnale della scoperta delle relazioni direttamente nella conversazione. Una seconda direzione è lo speech-to-text general-purpose per i media caricati. Una terza è un'harness di valutazione in stile TREC su un sottoinsieme etichettato del catalogo, che confronti la pipeline ibrida contro il puro recupero denso e il puro recupero per keyword sugli stessi chunk.

Da una prospettiva più ampia, l'architettura illustra un pattern ricorrente nei sistemi retrieval-augmented: quando un modello generativo è vincolato a operare esclusivamente su evidenza recuperata, le decisioni progettuali più consequenziali non riguardano il modello in sé ma la struttura del corpus e i segnali usati per ordinarlo. Lo schema a due livelli, la fusione pesata per prossimità e i boost graduali sono tutti raffinamenti di quella intuizione di base. Sono poco costosi da implementare, facili da ragionare e osservabili nelle risposte che producono.

Riferimenti Bibliografici

[1] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 4, pp. 824–836, 2020.

[2] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, 2019, pp. 3980–3990.

[3] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, "Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers," in *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, 2020, pp. 5776–5788.

Tabella III
RIEPILOGO DEI CONTRIBUTI ORIGINALI.

Contributo	Sezione	Descrizione
Schema documentale a due livelli	§II	Documenti riassunto e chunk separati per task di recupero ortogonali
Estrazione keyword RAKE per recupero ibrido	§III.B	Garantisce che le corrispondenze verbatim di entità nominate non siano mai perse
Pesatura gaussiana per prossimità su RRF	§III.C	Premia la co-locazione compatta delle keyword rispetto alla dispersione nella query
Prenotazione di slot per keyword	§III.E	Garantisce le corrispondenze keyword nell'output finale indipendentemente dal punteggio denso
Boost di copertura (frazione normalizzata)	§V.C	Segnale di copertura multi-documento normalizzato per lunghezza della sorgente
Citation boost graduale (saturazione esponenziale)	§V.D	Forza della citazione continua basata sulla frequenza di menzione
Aggregazione MAX per archi non direzionali del grafo	§V.F	Preserva i segnali di similarità asimmetrica nella scoperta
Acquisizione del vocabolario dalle relazioni curate	§V.G	Le conferme del curatore diventano addestramento in-context a ogni etichettatura
Generazione di etichette simmetriche con vocabolario	§V.H	Le relazioni non direzionali ricevono solo etichette agnostiche rispetto alla direzione
Auto-conferma in bulk con persistenza in streaming	§V.I	Singola chiamata LLM per etichettare tutti i suggerimenti pendenti; variante SSE per UI live
Ancoraggio rigoroso tramite shortcode di citazione inline	§IV.C	Risposte del chatbot completamente attribuibili e verificabili
Selezione del modello scope-aware (item / org / full)	§IV.B	Accoppia bersaglio di recupero e livello di accesso in un'unica superficie API; abilita la composi

[4] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," in *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, 2020, pp. 9459–9474.

[5] G. V. Cormack, C. L. A. Clarke, and S. Büttcher, "Reciprocal rank fusion outperforms condorcet and individual rank learning methods," in *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '09)*. New York, NY, USA: ACM, 2009, pp. 758–759.

[6] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[7] F. Al Machot, M. T. Horsch, and H. Ullah, "Building trustworthy ai: Transparent ai systems via large language models, ontologies, and logical reasoning (transpnet)," 11 2024. [Online]. Available: https://arxiv.org/abs/2411.08469

[8] M.-C. Dinu, C. Leoveanu-Condrei, M. Holzleitner, W. Zellinger, and S. Hochreiter, "Symbolicai: A framework for logic-based approaches combining generative models and solvers," 2 2024. [Online]. Available: https://arxiv.org/abs/2402.00854

[9] S. Rose, D. Engel, N. Cramer, and W. Cowley, "Automatic keyword extraction from individual documents," in *Text Mining: Theory and Applications*, M. W. Berry and J. Kogan, Eds. John Wiley & Sons, 2010, pp. 1–20.

[10] P. Tarau, "On llm-generated logic programs and their inference execution methods," *Electronic Proceedings in Theoretical Computer Science (EPTCS)*, vol. 416, pp. 1–14, 2 2025. [Online]. Available: https://arxiv.org/abs/2502.09209

[11] S. Zhu and S. Sun, "Exploring knowledge graph-based neural-symbolic system from application perspective," 2024. [Online]. Available: https://arxiv.org/abs/2405.03524

[12] Anthropic, "The Claude 3 model family: Opus, Sonnet, Haiku," Anthropic, Tech. Rep., 2024, model card. [Online]. Available: https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf