Costo di Caricamento e Indicizzazione di File e Media

Quando un utente vuole aggiungere documenti (PDF, DOC, immagini, video, ecc.) nella propria *Knowledge Box* (KBox) o *Vector Store* (MongoDB Atlas), l'applicazione esegue diversi **passi**:

1. **Processamento iniziale** del file (lettura, estrazione testo, eventuale OCR o captioning se è un'immagine/video).
2. **Chunking** (divisione in blocchi di lunghezza moderata per migliorare la ricerca).
3. **Calcolo degli embedding** dei chunk (con modello dedicato di embedding).
4. **Salvataggio** dei chunk (testo e metadata) nel DB e **salvataggio** dei relativi vettori embedding nel Vector Store.

Ciascuno di questi passi introduce un **costo**. Di seguito lo *parametrizziamo* e poi presentiamo un *esempio* per diversi tipi di file: PDF in pipeline Hi-Res, documento testuale standard (pipeline Fast), immagine e video.

1. Parametri e Formule Generali

Definiamo una **formula generale** per il caricamento di un file (o media) che abbia:

- N_{pages}: numero di pagine (se testo/PDF). Oppure, se immagine, consideriamo 1 "pagina"; se video, potrà essere equiparato a un certo numero di frame/pagine elaborate.
- $C_{\text{processing}}$: costo di *processing* "Unstructured" (o analogo) per estrarre il testo (oppure, nel caso di immagini/video, la parte di OCR/captioning).
- $C_{\text{embedding}}$: costo di generazione embedding per i chunk ottenuti.
- C_{DB}: costo di scrittura dei chunk e dei vettori embedding su MongoDB Atlas.

1.1 Costo di Processamento (Unstructured)

Assumiamo due pipeline tipiche:

1. **Fast Pipeline**: \$0,001 per pagina
2. **Hi-Res Pipeline**: \$0,01 per pagina

(Fonte: Unstructured.io, oppure AWS Marketplace equivalenti)

Formula di base

$$C_{\text{processing}} = N_{\text{pages}} \times c_{\text{page}},$$

dove $c_{\text{page}} \in \{0,001,\ 0,01\}$ a seconda di Fast o Hi-Res.

1.2 Chunking

Il file di testo (ottenuto da PDF, DOC, o da un OCR) viene suddiviso in blocchi di lunghezza $\bar{T}_{\text{chunk}}$ token. Se il testo complessivo di un file è T_{total} token, il **numero di chunk**:

$$N_{\text{chunk}} = \left\lceil \frac{T_{\text{total}}}{\bar{T}_{\text{chunk}}} \right\rceil.$$

(In alcuni casi, 1 chunk per pagina se ∼500 token/pagina.)

1.3 Calcolo degli Embedding

Per ogni chunk si genera un embedding (es. con *text-embedding-3-small*).

- **Costo** per 1k token = \$0,00002 (o meno).

Se un chunk tipico ha $\bar{T}_{\text{chunk}}$ token, allora il **costo embedding** di 1 chunk:

$$C_{\text{embed\_chunk}} = \frac{\bar{T}_{\text{chunk}}}{1000} \times c_{\text{embed}}.$$

Moltiplicando per N_{chunk}:

$$C_{\text{embedding}} = N_{\text{chunk}} \times C_{\text{embed\_chunk}}.$$

1.4 Costo di Scrittura DB e Vector Store

- Ogni chunk (testo + metadata) si inserisce in una collezione DB, con un costo di scrittura $\approx d3$.
- Ogni embedding (es. 1536 dimensioni) occupa $\sim$ 6KB: scrittura su Vector Store con $\approx$ 6 WPU.

$$C_{\text{DB\_1\_chunk}} = \underbrace{\frac{1}{10^6} \times C_{\text{WPU}}}_{\text{testo}} + \underbrace{\frac{\text{embedding\_KB}}{10^6} \times C_{\text{WPU}}}_{\text{vettore}}$$

Se embedding_KB ≈ 6 e $C_{\text{WPU}} \approx 1,25$ \$, allora scrivere 1 chunk con embedding costerà $\approx 7{,}5 \times 10^{-6}$\$.
Per semplicità:

$$C_{\text{DB}} = N_{\text{chunk}} \times C_{\text{DB\_1\_chunk}}.$$

1.5 Formula di Caricamento di un File

Combinando il tutto:

$$\boxed{C_{\text{upload\_file}} = C_{\text{processing}} + C_{\text{embedding}} + C_{\text{DB}}}$$

dove:

$$C_{\text{processing}} = N_{\text{pages}} \times c_{\text{page}},$$

$$N_{\text{chunk}} = \left\lceil \frac{T_{\text{total}}}{\bar{T}_{\text{chunk}}} \right\rceil,$$

$$C_{\text{embedding}} = N_{\text{chunk}} \times \frac{\bar{T}_{\text{chunk}}}{1000} c_{\text{embed}},$$

$$C_{\text{DB}} = N_{\text{chunk}} \times C_{\text{DB\_1\_chunk}}.$$

2. Caso di Documenti PDF (Hi-Res vs. Standard)

2.1 Caso "PDF con Pipeline Hi-Res"

- $c_{\text{page}} = 0,01\$(\textit{Hi-Res}).\textit{Esempio}$:$\text{N}_{\text{pages}} = 10$.

$$C_{\text{processing}} = 10 \times 0{,}01 = 0{,}10\$ \quad (= 10 \text{ cent totali}).$$

Se ogni pagina $\sim$ 500 token, allora $T_{\text{total}} = 5000$ token su 10 pagine, e con chunk da 500 token: $N_{\text{chunk}} = 10$.

Embedding

$$C_{\text{embed\_chunk}} = \frac{500}{1000} \times 0{,}00002 = 0{,}00001\$ \quad (1 \times 10^{-5}).$$

Dieci chunk $\rightarrow 10 \times 10^{-5} = 10^{-4}$\$.

DB Scrivere 1 chunk con embedding $\approx 7,5 \times 10^{-6}\$; per 10 chunk$:7,5×10^{-5}\$.

$$C_{\text{upload\_file}} = 0{,}10 + 0{,}0001 + 0{,}000075 \approx 0{,}100175 \approx 0{,}10\$.$$

Quasi tutto dovuto alla pipeline Hi-Res.

2.2 Caso "Documento Standard" (Fast Pipeline)

- $c_{\text{page}} = 0{,}001\$(Fast). Esempio : 10 pagine \rightarrow 10 \times 0,001 = 0,01\$ diprocessing.$

Stessa logica di embedding e DB: $0,0001 + 0,000075 = 0,000175\$$.

$$C_{\text{upload\_file}} \approx 0{,}01 + 0{,}000175 = 0{,}010175 \approx 0{,}0102\$.$$

3. Caso di File Multimediali (Immagini e Video)

Immagini: potremmo usare GPT-4o (o GPT-4o Mini) per ricavare una descrizione testuale ("caption"). L'immagine diventa input visivo, convertito in token.
Video: estraiamo alcuni frame, ognuno trattato come un'immagine.

3.1 Caricamento di un'Immagine

Passi:

1. **Caption**: chiamata LLM in input con $\approx T_{\text{img}}$ token (dipende da dimensione, es. 512×512 $\rightarrow$ 255 token).
2. **Output**: testo di descrizione (es. 50–100 token).
3. **Embedding**: dei $\sim$ 100 token descrizione.
4. **Scrittura**: salviamo descrizione e embedding nel DB.

Formula

$$C_{\text{upload\_img}} = C_{\text{LLM\_caption}} + C_{\text{embed}} + C_{\text{DB}},$$

dove

$$C_{\text{LLM\_caption}} = \frac{T_{\text{img}}}{1000}\, p_{\text{in}} + \frac{T_{\text{descr}}}{1000}\, p_{\text{out}},$$

$$C_{\text{embed}} \approx \frac{T_{\text{descr}}}{1000} \times c_{\text{embed}}, \quad C_{\text{DB}} \approx 7,5 \times 10^{-6}\$ \text{ (per embedding da 6KB)}.$$

Esempio (con GPT-4o)

$$T_{\text{img}} = 255, \quad T_{\text{descr}} = 100, \quad p_{\text{in}} = 0{,}005, \quad p_{\text{out}} = 0{,}015.$$

$$C_{\text{LLM\_caption}} = \frac{255}{1000} \times 0{,}005 + \frac{100}{1000} \times 0{,}015 = 0{,}001275 + 0{,}0015 = 0{,}002775\$.$$

Embedding 100 token $\rightarrow 2 \times 10^{-6}\$; DB scrittura \sim 7,5 \times 10^{-6}\$. Totale \approx 0{,}00278\$(\sim 0,28 cent)$.

(Se GPT-4o Mini), costi scendono di 30×, con input 255×0,00015 + output 100×0,00060 $\approx 0,00009825\$$.

3.2 Caricamento di un Video

Passi:

1. **Estrazione Frame**: supponiamo di estrarre 1 frame ogni X secondi, generando N_{frame} immagini.
2. **Caption** per ciascun frame, come se fosse un'immagine.
3. **Embedding**: potremmo unire le caption in un testo unico, generando un embedding, o creare embedding per ogni frame.
4. **Scrittura**: su DB.

Formula

$$C_{\text{video}} \approx N_{\text{frame}} \times C_{\text{LLM\_frame}} + C_{\text{embed}} + C_{\text{DB}}.$$

Esempio

- Video 2 min, estraiamo 1 frame/10 s $\rightarrow N_{\text{frame}} = 12$.
- GPT-4o per caption di ogni frame (ipotizziamo $\sim$ 0,0027\$)$\rightarrow$ 12×0,0027 = 0,0324\$.$Testototale12caption\times$ 50 token = 600 token $\rightarrow$ embedding = 600/1000 × 0,00002 = $1{,}2 \times 10^{-5}$\$.

- DB scrittura embedding $\sim 7{,}5\times10^{-6}$\$. Totale $\approx$ 0,032417\$($\sim 3,24cent$).**(Se GPT-4o Mini)**, $costiscendonodiunfattore$ -30.

4. Conclusioni ed Esempi Riassuntivi

Caricamento di documenti (testuali):

- *Hi-Res* (\$0,01/pagina) domina i costi (\$0,10 per 10 pagine).
- *Fast* (\$0,001/pagina) scende a millesimi.

Caricamento di immagini:

- Principale costo se usiamo GPT-4o per caption (fino a 0,002–0,003\$ per immagine).
- Embedding e DB microcentesimi.

- GPT-4o Mini: $\sim 0,0001\$perimmagine$.
  Caricamento di video:
  - Diviso in "frame extraction + captioning" (nessun cenno ad altro).
  - Esempio 2 min, 12 frame, GPT-4o $\approx 0,0324\$tot(caption).EmbeddingeDBtrascurabili.GPT-4oMiniriducedi$ 20×.

Formula Generale per un file (testo o media) caricabile:

$$\boxed{C_{\text{upload\_file}} = C_{\text{process}} + N_{\text{chunk}} \times \left[\frac{\bar{T}_{\text{chunk}}}{1000}\, c_{\text{embed}} + C_{\text{DB\_chunk}}\right] + (\text{costo extra se immagine/video}).}$$

Dove

- $C_{\text{process}} = N_{\text{pages}} \times c_{\text{page}}$ (0,001\$ o 0,01\$).
- $N_{\text{frame}} = \frac{\text{durata\_s}}{\text{sampling\_sec}}$ per i video.
- $C_{\text{LLM\_caption}} = \left(\frac{T_{\text{img}}}{1000}\right)p_{\text{in}} + \left(\frac{T_{\text{descr}}}{1000}\right)p_{\text{out}}$ per ogni immagine/frame.

Esempio Finale Riassuntivo

Tipo File	Pagine/Frame	Pipeline/Modello	Costo Process.	Costo LLM
PDF 10 pag (Hi-Res)	10	0,01\$/pagina	0,10\$	–
PDF 10 pag (Fast)	10	0,001\$/pagina	0,01\$	–
Immagine (GPT-4o)	1	$(p_{\text{in}} = 0,005; p_{\text{out}} = 0,015)$	–	$\sim 0,0028$\$
Immagine (Mini)	1	$(p_{\text{in}} = 0,00015; p_{\text{out}} = 0,00060)$	–	$\sim 0,0001$\$
Video 2 min (GPT-4o)	12 frame	$\sim 0,0027$\$ x frame	–	$12 \times 0,0027 = 0,$
Video 2 min (Mini)	12 frame	$\sim 0,0001$\$ x frame	–	$12 \times 0,0001 = 0,$

Come si vede:

- *PDF* con pipeline Hi-Res: \$0,10…\$0,20 in base al numero di pagine (qui 10).

- *PDF* con pipeline Fast: 10× meno.
- *Immagini e video*: costo imputabile quasi tutto alle *chiamate LLM* (GPT-4o o GPT-4o Mini) per ottenere le descrizioni. Embedding + DB restano microcentesimi.

In conclusione, il *costo di caricamento* di file (testo/immagini/video) è spesso dominato:

- Dai **cost per pagina** (se si usa SaaS Unstructured).
- Dalle **chiamate GPT-4o** (se facciamo caption su immagini/video).
- L'**embedding** e DB incidono in microcentesimi.

Pertanto, un PDF grande con pipeline Hi-Res rimane su alcuni centesimi (0,01\$ × pagine), mentre un PDF standard (Fast) scende a millesimi. Per le immagini e i video *dipende dal modello LLM* usato per la descrizione: GPT-4o può costare $\sim 0,003$\$/immagine, GPT-4o Mini $\sim 0,0001$\$.
Lo **storage** finale su MongoDB inciderà poi su base mensile (0,25\$/GB), se l'utente carica molti MB/GB di contenuti multimediali.