7. Information Retrieval

Il recupero dell'informazione (IR) consiste nel trovare materiale, generalmente documenti di natura non strutturata (solitamente testo), che soddisfi un bisogno informativo all'interno di grandi collezioni, di solito archiviate su computer.

Oggi si pensa immediatamente alla ricerca sul web, ma esistono molti altri contesti in cui il recupero dell'informazione è essenziale. Alcuni esempi includono:

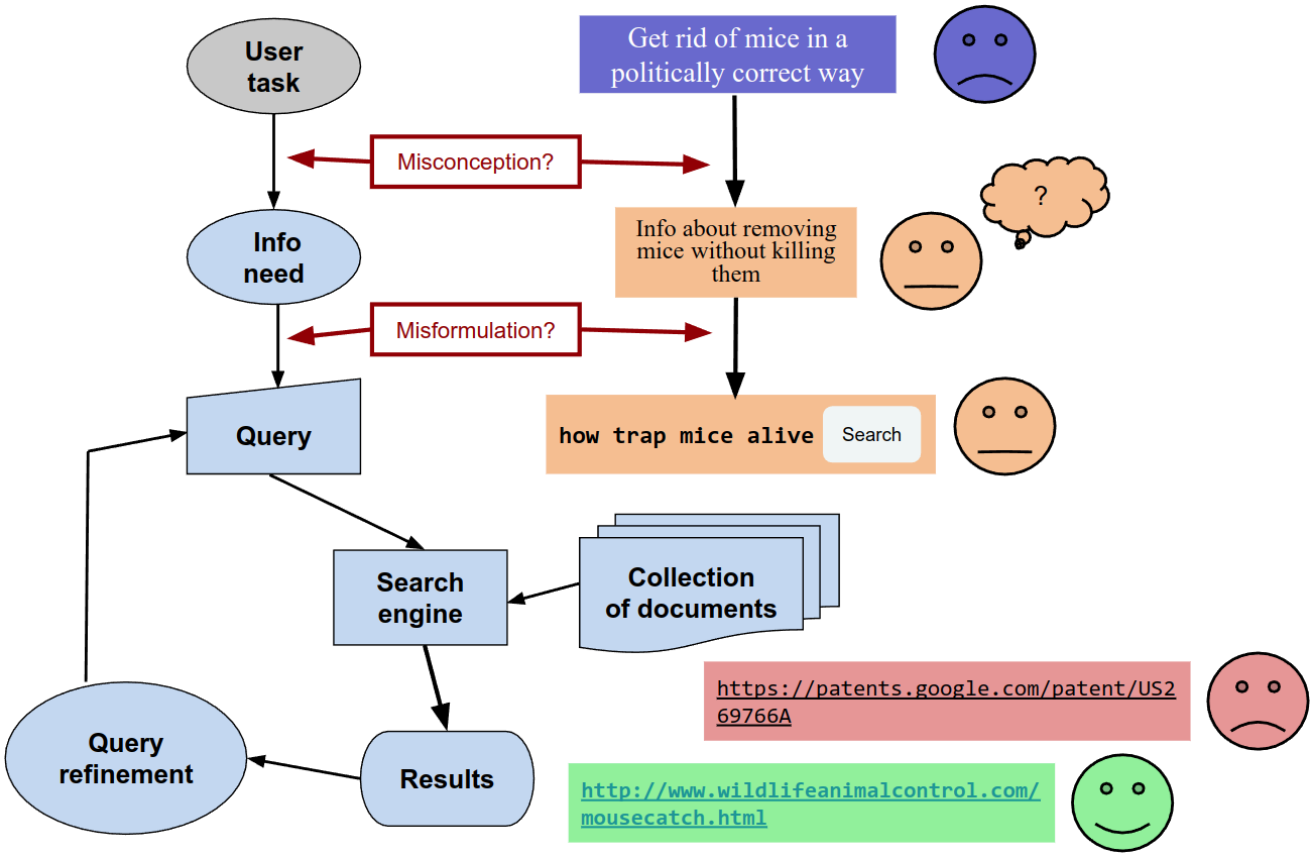
- La ricerca in e-mail e social media
- La ricerca di file sul proprio laptop
- Le basi di conoscenza aziendali
- La ricerca in domini specifici, come il recupero di informazioni legali
- Le biblioteche digitali
- La ricerca di notizie

Assunzioni di base

Si considerano alcune assunzioni fondamentali nel recupero dell'informazione:

- Un **documento** è un file non strutturato.
- Una **collezione** è un insieme di documenti.
- Si assume, per il momento, che la collezione sia statica e priva di ipertesto.
- Una **query** è un insieme di parole chiave utilizzato per esprimere un bisogno informativo.
- L'**obiettivo** è recuperare documenti che contengano informazioni rilevanti per il bisogno informativo dell'utente e che lo aiutino a completare un determinato compito.
-

Modello di ricerca classico



Cos'è un documento?

Un documento può assumere molte forme diverse. Alcuni esempi includono: pagine web, e-mail, libri, articoli di notizie, pubblicazioni scientifiche, messaggi di testo, documenti Word, presentazioni PowerPoint, file PDF, post nei forum, brevetti, sessioni di messaggistica istantanea e molto altro.

I documenti condividono alcune proprietà comuni. In generale, contengono una quantità significativa di testo e sono per lo più non strutturati. Tuttavia, spesso presentano alcuni elementi strutturati, come il titolo, l'autore e la data nel caso degli articoli, oppure il soggetto, il mittente e il destinatario per le e-mail.

Recupero dei dati vs recupero dell'informazione

Un sistema di **recupero dei dati** (Data Retrieval), come un DBMS, gestisce dati con una struttura ben definita, organizzati secondo uno schema di database. Al contrario, un sistema di **recupero dell'informazione** (Information Retrieval) opera su testi scritti in linguaggio naturale, spesso non strutturati e ambigui, come i documenti testuali.

Differenze principali

Un **record di database** presenta un contenuto strutturato in campi tipizzati, dove i valori rispettano i tipi di dati definiti e possono includere codici, date o prezzi. I dati sono archiviati in formati predefiniti.

Un **documento testuale**, invece, è generalmente non strutturato e può contenere qualsiasi tipo di informazione espressa in linguaggio naturale.

Un sistema di **recupero dei dati** utilizza un linguaggio di interrogazione formalmente definito, come SQL o l'algebra relazionale.

Un sistema di **recupero dell'informazione** recupera documenti tramite parole chiave (query o ricerca) espresse in linguaggio naturale.

Differenze tra linguaggi di interrogazione e ricerca

Il **linguaggio di interrogazione** si basa su una grammatica formale e produce sempre un insieme prevedibile di record che soddisfano esattamente la query, indipendentemente dall'implementazione del sistema.

Il **linguaggio di ricerca** si basa su parole chiave del linguaggio naturale e può essere interpretato in modi diversi a seconda del sistema IR. Il risultato non è un insieme esatto di documenti, ma una lista ordinata in base alla rilevanza rispetto alla query.

Confronto tra testi

Il cuore del recupero dell'informazione consiste nel confrontare il testo della query con il testo dei documenti e determinare quali siano le corrispondenze migliori.

Il semplice confronto basato sulla corrispondenza esatta delle parole non è sufficiente, poiché in un linguaggio naturale come l'inglese esistono molti modi diversi per esprimere lo stesso concetto.

Ad esempio, una notizia che contiene la frase "il direttore di banca ad Amherst ruba fondi" dovrebbe corrispondere a una query che cerca informazioni su furti in banca? Alcuni articoli saranno più pertinenti di altri.

Per questo motivo, è necessario un concetto di **rilevanza** piuttosto che una semplice corrispondenza esatta. I diversi modelli di recupero delle informazioni si basano su differenti definizioni di rilevanza.

Modelli di recupero

Esistono diversi modelli di recupero delle informazioni, ognuno con approcci e criteri di rilevanza differenti. I **modelli classici** includono:

- Il **modello booleano**, che si basa su operatori logici per determinare la corrispondenza tra documenti e query.
- Il **modello dello spazio vettoriale**, che rappresenta documenti e query come vettori in uno spazio multidimensionale, calcolando la similarità tramite misure come il coseno dell'angolo tra i vettori.

- I **modelli probabilistici**, che stimano la probabilità che un documento sia rilevante rispetto a una query.

Un esempio avanzato di modello probabilistico è **BM25 (Best Match 25)**, una delle tecniche più efficaci per il ranking dei documenti in base alla loro pertinenza.

Un'altra categoria è quella dei **modelli linguistici**, che utilizzano modelli statistici per rappresentare la distribuzione delle parole nei documenti e nelle query. Oltre a questi approcci, esistono metodi che combinano diverse fonti di evidenza per migliorare l'efficacia del recupero. Tra questi troviamo:

- Le **reti inferenziali**, che modellano la relazione tra query e documenti come una rete di dipendenze probabilistiche.
- Le tecniche di **Learning to Rank**, che sfruttano algoritmi di apprendimento automatico per ordinare i risultati in base alla loro rilevanza.

I progressi nei modelli di recupero delle informazioni hanno portato a miglioramenti significativi nell'efficacia della ricerca.

Modello di recupero booleano

Il modello di recupero booleano permette di formulare query sotto forma di espressioni booleane.

Le ricerche booleane utilizzano gli operatori logici **AND, OR e NOT** per combinare i termini di ricerca. Ogni documento viene visto come un insieme di parole (**Bag-of-Words**) e la corrispondenza è precisa: un documento soddisfa esattamente la condizione della query oppure no.

Questo modello è uno dei più semplici per costruire un sistema di recupero dell'informazione. Per circa tre decenni, fino all'inizio degli anni '90 e alla diffusione del World Wide Web, è stato lo strumento principale per il recupero commerciale delle informazioni.

Ancora oggi, molti sistemi di ricerca utilizzano un approccio booleano, come la ricerca nelle e-mail, nei cataloghi delle biblioteche o nello strumento **Spotlight** di Mac OS X.

Esempio di ricerca booleana

Si consideri la query: **quali opere di Shakespeare contengono le parole "Brutus" e "Caesar" ma non "Calpurnia"?**

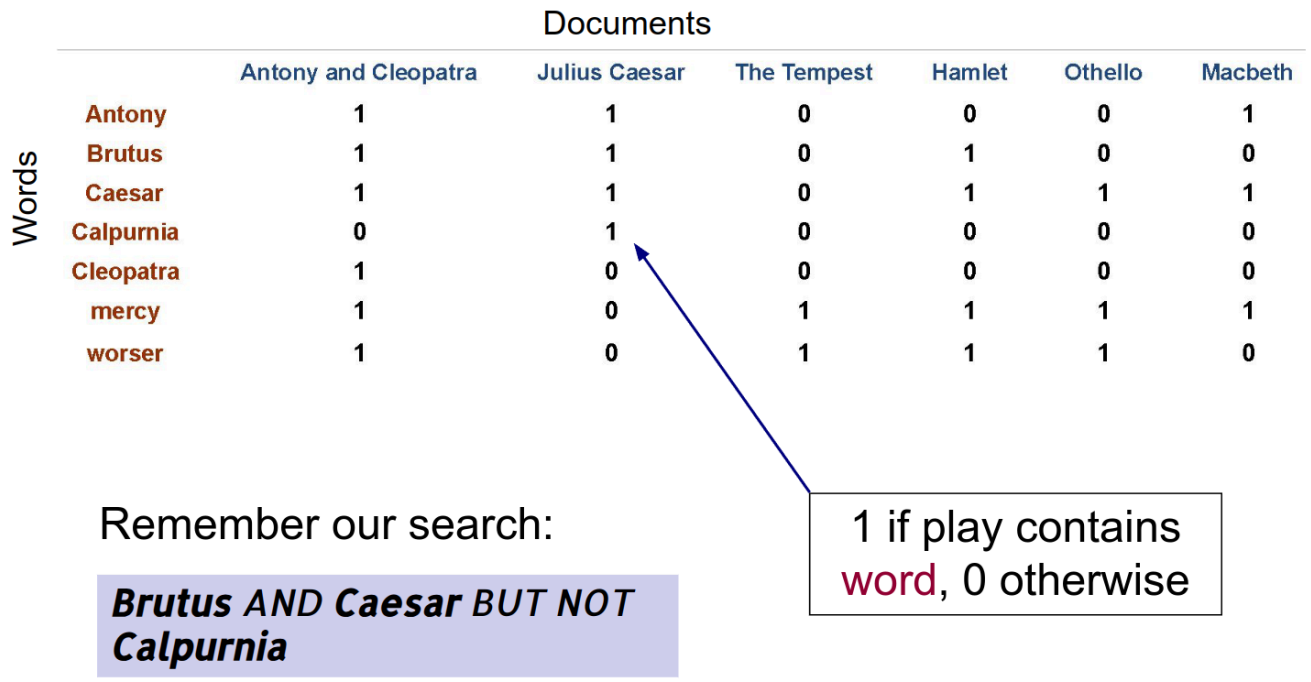
Un possibile approccio potrebbe essere quello di cercare nei testi di Shakespeare tutte le occorrenze di "Brutus" e "Caesar", per poi escludere le righe che contengono "Calpurnia".

Tuttavia, questo metodo presenta alcuni problemi:

- È **lento** per grandi raccolte di documenti.

- L'operazione **NOT Calpurnia** non è banale da eseguire, poiché bisogna rimuovere non solo righe, ma interi documenti che contengono la parola.
- È necessario un **modello strutturato** che rappresenti i documenti come insiemi di parole, per facilitare la ricerca e il filtraggio.

Nonostante i suoi limiti, il modello booleano è ancora ampiamente utilizzato in specifici contesti dove è richiesta precisione assoluta nelle corrispondenze.



Vettori di incidenza

Nel modello booleano, ogni parola è rappresentata da un vettore binario (0/1), detto **vettore di incidenza**, in cui ogni bit indica la presenza (1) o l'assenza (0) della parola in un determinato documento.

Esempio. Per rispondere a una ricerca, si prendono i vettori di incidenza delle parole richieste e si applicano operazioni bitwise. Ad esempio, per la query che cerca i documenti contenenti **Brutus** e **Caesar**, ma non **Calpurnia**, si procede come segue:

1. Si recuperano i vettori di incidenza delle parole "Brutus", "Caesar" e "Calpurnia".
2. Si complementa il vettore di "Calpurnia" (ovvero si invertono 0 e 1).
3. Si applica l'operazione **AND bitwise** tra i vettori:

$$110100 \quad (\text{Brutus}),\ 110111 \quad (\text{Caesar})\ 101111 \quad (\text{Calpurnia, complementato})$$

$$110100\ \text{AND}\ 110111\ \text{AND}\ 101111 = 100100$$

Il risultato è un nuovo vettore che indica in quali documenti la query ha una corrispondenza esatta.

Questa rappresentazione consente un'implementazione efficiente delle ricerche booleane utilizzando operazioni logiche veloci a livello di bit.

	Answer:	1 Antony and Cleopatra	0 Julius Caesar	0 The Tempest	1 Hamlet	0 Othello	0 Macbeth
Words	Antony	1	1	0	0	0	1
	Brutus	1	1	0	1	0	0
	Caesar	1	1	0	1	1	1
	Calpurnia	0	1	0	0	0	0
	Cleopatra	1	0	0	0	0	0
	mercy	1	0	1	1	1	1
	worser	1	0	1	1	1	0

Dimensione della matrice

Consideriamo un caso in cui il numero totale di documenti sia $N = 1$ milione, con ogni documento contenente circa 1000 parole. Se consideriamo una media di **6 byte per parola** (includendo spazi e punteggiatura), i documenti occupano complessivamente **6GB di dati**.

Supponiamo che il numero di termini distinti sia $M = 500.000$. In questo caso, la matrice di incidenza avrà dimensioni $M \times N = 500.000 \times 1.000.000$, ovvero mezzo trilione di valori binari (0 e 1). Tuttavia, questa matrice contiene **non più di un miliardo di 1**, il che significa che è **estremamente sparsa** (cioè, la maggior parte degli elementi è 0).

Rappresentazione più efficiente

Dato che la matrice è molto grande e contiene prevalentemente zeri, una memorizzazione diretta non è efficiente. Un'alternativa migliore è **registrare solo le posizioni degli 1**, riducendo drasticamente lo spazio necessario per rappresentare la matrice. Questo metodo permette di gestire grandi collezioni di documenti in modo più scalabile ed efficiente.

Indice invertito

Per ottimizzare la ricerca nei documenti, si utilizza una struttura chiamata **indice invertito**. In questo modello, per ogni termine t viene memorizzata una lista di tutti i documenti che lo contengono.

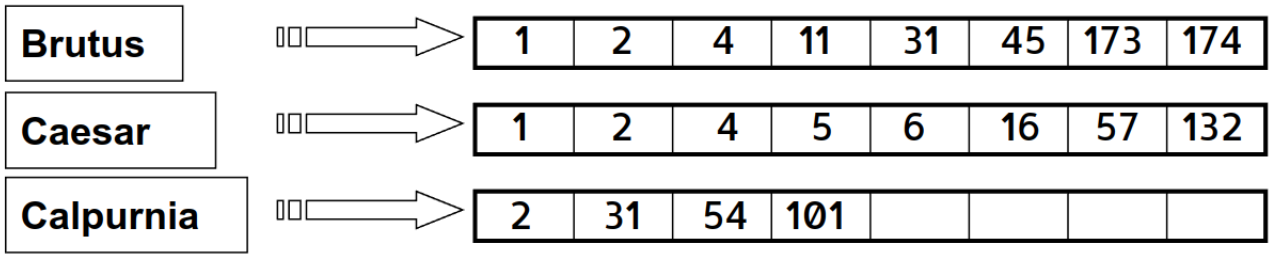
Ogni documento viene identificato tramite un **docID**, ovvero un numero seriale univoco.

Rappresentazione dell'indice

Un'idea potrebbe essere quella di usare **array a dimensione fissa** per ogni termine, ma questo approccio presenta problemi di scalabilità. Infatti, se una nuova parola viene aggiunta a un documento, potrebbe essere necessario ridimensionare l'array, causando inefficienza nella gestione della memoria.

Ad esempio, se il termine **"Caesar"** viene aggiunto al **documento 14**, dovremmo aggiornare la lista dei documenti contenenti quel termine. Se la lista fosse implementata con array statici, bisognerebbe riallocarla per includere il nuovo documento, un'operazione costosa.

Per questo motivo, gli **indici invertiti** utilizzano generalmente **liste dinamiche** o strutture come alberi bilanciati o tabelle hash per garantire aggiornamenti efficienti e un recupero rapido delle informazioni.



Liste di occorrenze (Posting-lists)

Per ogni termine in un indice invertito, è necessario memorizzare una **lista di occorrenze** (**posting-list**) che contenga i documenti in cui quel termine appare. Dato che il numero di documenti per ciascun termine varia, queste liste devono avere **dimensioni variabili**.

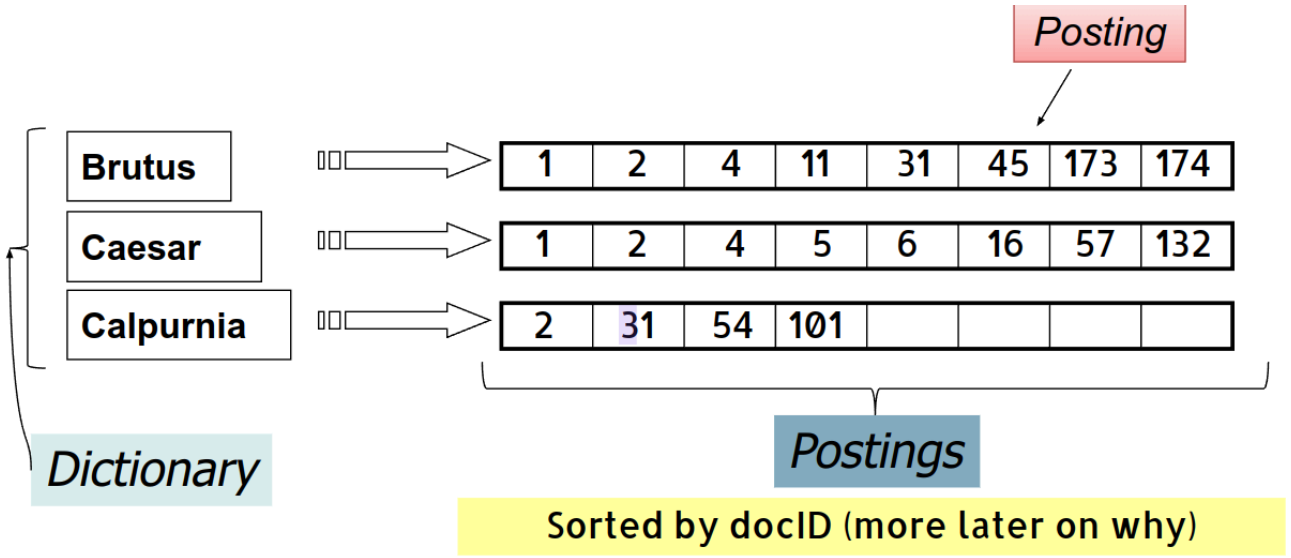
Memorizzazione delle liste di occorrenze

- **Su disco**: è preferibile memorizzare le liste come blocchi contigui di dati, poiché ciò migliora l'efficienza delle operazioni di lettura sequenziale.
- **In memoria principale**: si possono utilizzare **liste collegate** o **array a lunghezza variabile**, ognuno con vantaggi e svantaggi.

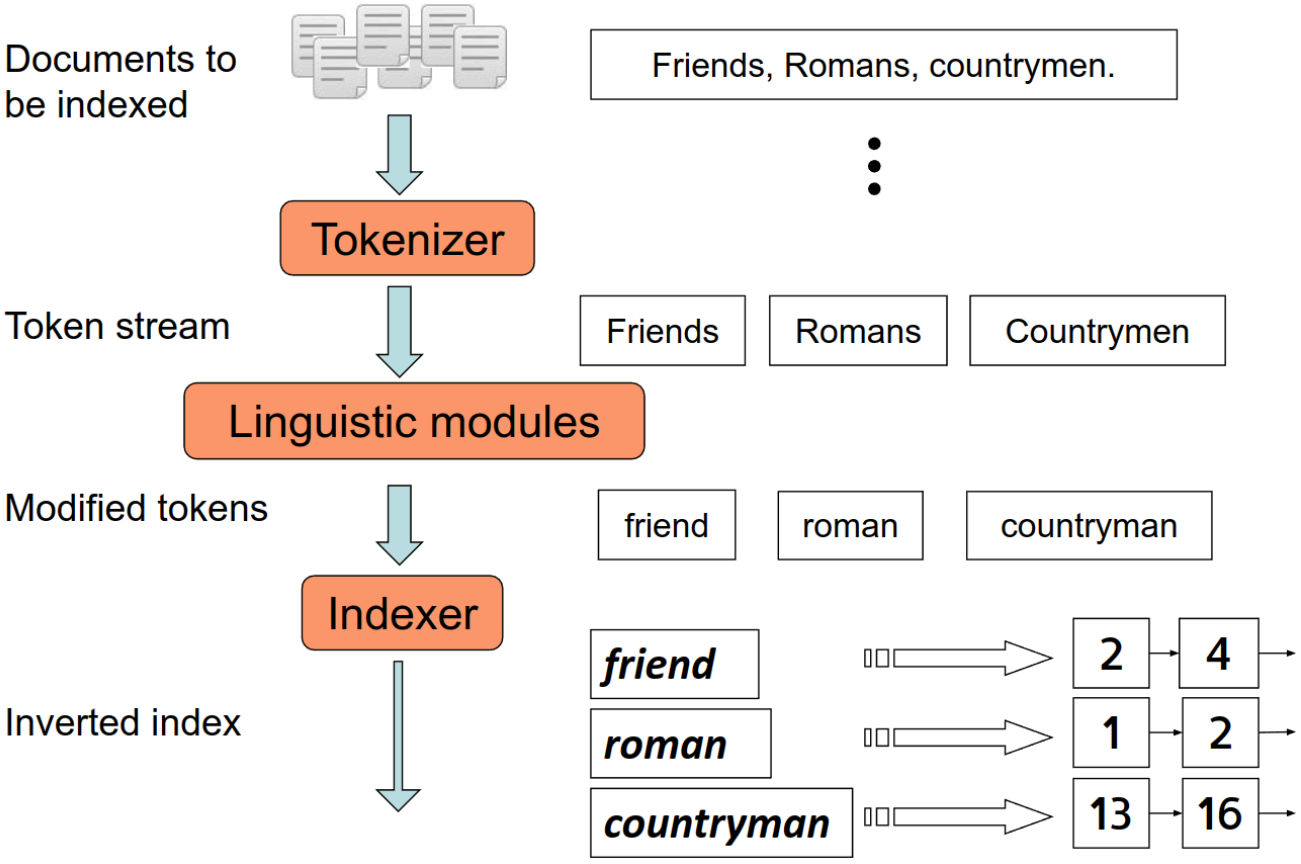
Compromessi tra spazio ed efficienza

L'uso di array variabili riduce l'overhead di gestione ma rende più complesso l'inserimento di nuovi elementi. Le liste collegate, invece, facilitano gli aggiornamenti dinamici ma introducono un maggiore consumo di memoria dovuto ai puntatori aggiuntivi.

La scelta della rappresentazione più efficiente dipende dal contesto d'uso e dai vincoli di spazio ed elaborazione.



Costruzione degli indici invertiti



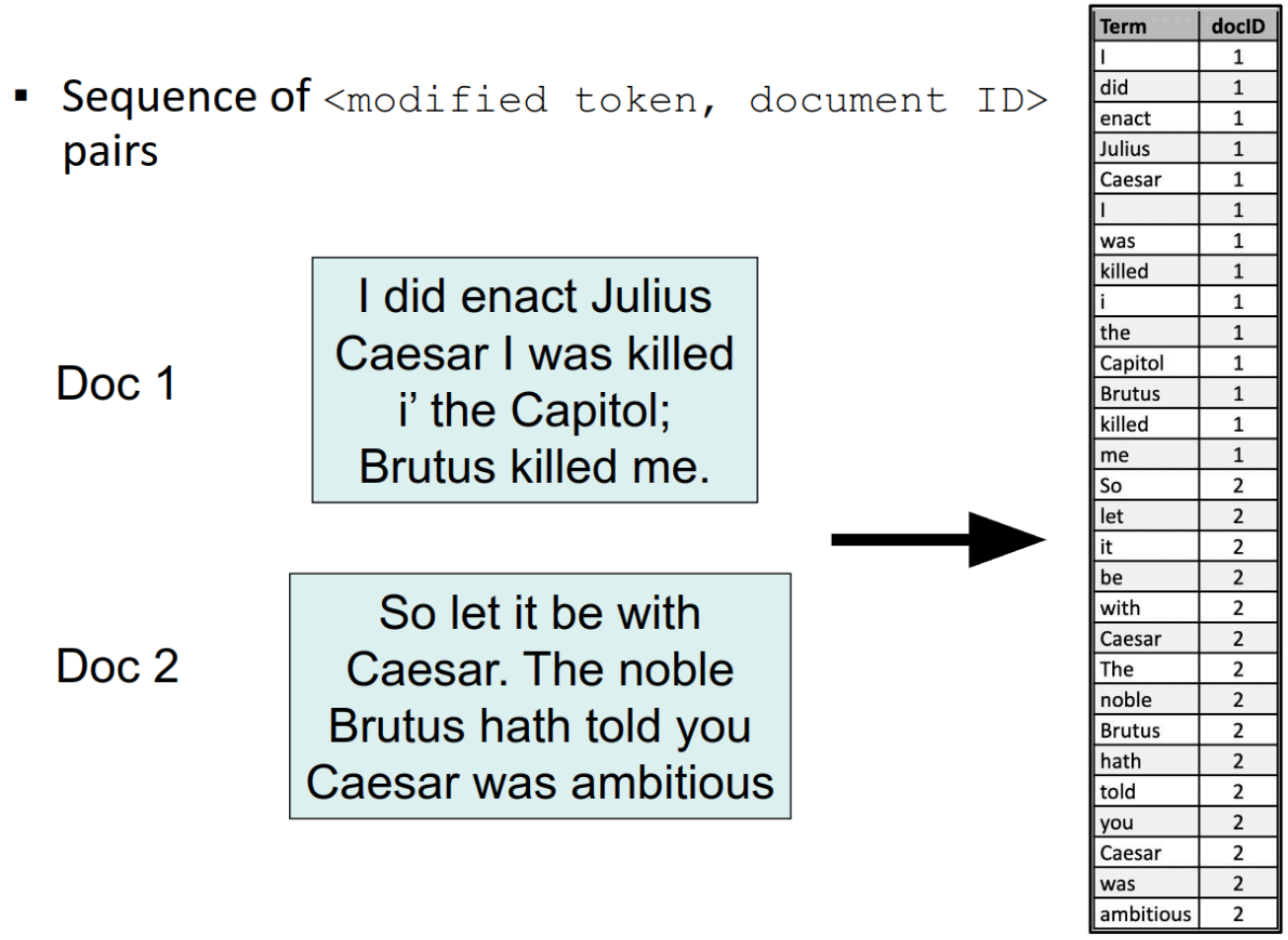
Pre-elaborazione del testo

Prima di poter effettuare una ricerca efficace nei documenti, è necessario eseguire una serie di operazioni di **pre-elaborazione del testo** per standardizzare e ottimizzare i dati.

Tokenizzazione

La **tokenizzazione** consiste nel suddividere una sequenza di caratteri in **token**, ovvero unità linguistiche come parole o frasi. Durante questo processo, è necessario gestire

correttamente casi particolari come **"John's"** o espressioni complesse come **"state-of-the-art"**, che potrebbero essere interpretate come un unico termine o più token separati.



Term	docID
I	1
did	1
enact	1
Julius	1
Caesar	1
I	1
was	1
killed	1
i	1
the	1
Capitol	1
Brutus	1
killed	1
me	1
So	2
let	2
it	2
be	2
with	2
Caesar	2
The	2
noble	2
Brutus	2
hath	2
told	2
you	2
Caesar	2
was	2
ambitious	2

Normalizzazione dei termini

Nel recupero dell'informazione, è spesso necessario **normalizzare** le parole sia nei testi indicizzati che nelle query, in modo da garantire corrispondenze più efficaci tra termini simili. L'obiettivo è far sì che varianti di una parola vengano considerate equivalenti. Ad esempio, vogliamo che **"U.S.A."** e **"USA"** siano trattati come lo stesso termine.

Un **termine** è una parola normalizzata che viene registrata nel dizionario del sistema IR. L'equivalenza tra termini può essere definita implicitamente tramite varie operazioni di normalizzazione, tra cui:

- **Eliminazione dei punti**: ad esempio, trasformare **"U.S.A."** in **"USA"**.
- **Eliminazione dei trattini**: ad esempio, considerare **"anti-discriminatory"** e **"antidiscriminatory"** come lo stesso termine.
- **Conversione in minuscolo**: ridurre tutte le lettere a minuscolo per evitare distinzioni non necessarie.

Ci sono alcuni casi in cui il maiuscolo ha un significato distintivo. Ad esempio:

- **General Motors** (nome proprio) non dovrebbe essere confuso con "general motors" (motori generali).

- **"Fed"** (Federal Reserve) è diverso da **"fed"** (forma verbale di "to feed").
- **"SAIL"** (nome proprio) e **"sail"** (vela) potrebbero avere significati diversi.

Tuttavia, nella pratica, la soluzione più comune è trasformare tutto in minuscolo, poiché gli utenti tendono a scrivere le query senza rispettare la corretta capitalizzazione.

Stemming

Lo **stemming** è il processo di riduzione dei termini alle loro radici prima dell'indicizzazione, eliminando suffissi e altre variazioni morfologiche.

Il termine "stemming" suggerisce un metodo **grezzo di troncamento dei suffissi**, che può risultare approssimativo ma è efficace per ridurre le variazioni linguistiche. Questo processo è **dipendente dalla lingua**, poiché le regole per il taglio dei suffissi variano tra le diverse lingue.

Ad esempio, le parole **"automate(s)", "automatic"** e **"automation"** possono essere tutte ridotte alla radice **"automat"**.

Un altro esempio è il trattamento dei termini **"compressed"** e **"compression"**, che possono essere normalizzati in **"compress"**, permettendo al sistema di recuperarli come equivalenti.

Tuttavia, esistono casi problematici in cui la riduzione può portare a errori, come nel caso di parole che vengono troncate eccessivamente o che finiscono per essere considerate erroneamente equivalenti.

Lemmatizzazione

La **lemmatizzazione** è il processo di riduzione delle forme flesse o varianti di una parola alla loro **forma base** o **lemma**, ovvero la voce principale del dizionario.

A differenza dello stemming, che utilizza regole semplici per tagliare i suffissi, la lemmatizzazione esegue una riduzione più accurata, basandosi sulla grammatica e sul significato delle parole.

- **"am", "are", "is" → "be"**
- **"car", "cars", "car's", "cars'" → "car"**

Un esempio più complesso mostra come la lemmatizzazione trasformi una frase intera in una forma normalizzata:

Frase originale:
"The boy’s cars are different colors."

Dopo la lemmatizzazione:
"the boy car be different color"

Differenza tra stemming e lemmatizzazione

Lo **stemming** si limita a troncare parti della parola senza considerare il contesto, mentre la **lemmatizzazione** effettua una riduzione più sofisticata, riportando ogni termine alla sua forma radice corretta, come appare in un dizionario. Per questo motivo, la lemmatizzazione è più accurata ma anche più computazionalmente costosa rispetto allo stemming.

Stop words

Le **stop words** sono parole molto comuni che, in alcuni sistemi di recupero dell'informazione, vengono completamente escluse dal dizionario. L'idea di eliminare le stop words si basa su due osservazioni principali:

- **Hanno un basso contenuto semantico**: parole come **"the", "a", "and", "to", "be"** contribuiscono poco al significato effettivo di una query.
- **Sono estremamente frequenti**: le 30 parole più comuni rappresentano circa **il 30% di tutte le occorrenze** in un corpus testuale.

Negli ultimi anni, si è osservata una riduzione dell'uso delle stop lists per diversi motivi:

- **Tecniche di compressione efficienti** rendono lo spazio necessario per memorizzare le stop words trascurabile.
- **Ottimizzazioni nelle query** permettono di gestire le stop words senza impattare significativamente i tempi di ricerca.

Ci sono situazioni in cui eliminare le stop words può compromettere la qualità della ricerca:

- **Query frasali**: ad esempio, nella ricerca di **"King of Denmark"**, rimuovere **"of"** altererebbe il significato della query.
- **Titoli di canzoni, citazioni famose, espressioni idiomatiche**: esempi come **"Let it be"** o **"To be or not to be"** perderebbero il loro senso senza le stop words.
- **Query relazionali**: una ricerca come **"flights to London"** dipende dal termine **"to"**, che indica la direzione del volo.

Per questi motivi, i moderni sistemi di recupero delle informazioni tendono a **conservare le stop words**, invece di eliminarle completamente.