Recuperación de información

Rubén Francisco Manrique
rf.manrique@uniandes.edu.co

Recuperación de Información

- La recuperación de información (IR) consiste en ***encontrar material*** (generalmente documentos) de naturaleza ***no estructurada*** (generalmente texto) que satisface ***una necesidad de información*** dentro de ***grandes colecciones*** (generalmente almacenadas en computadoras).

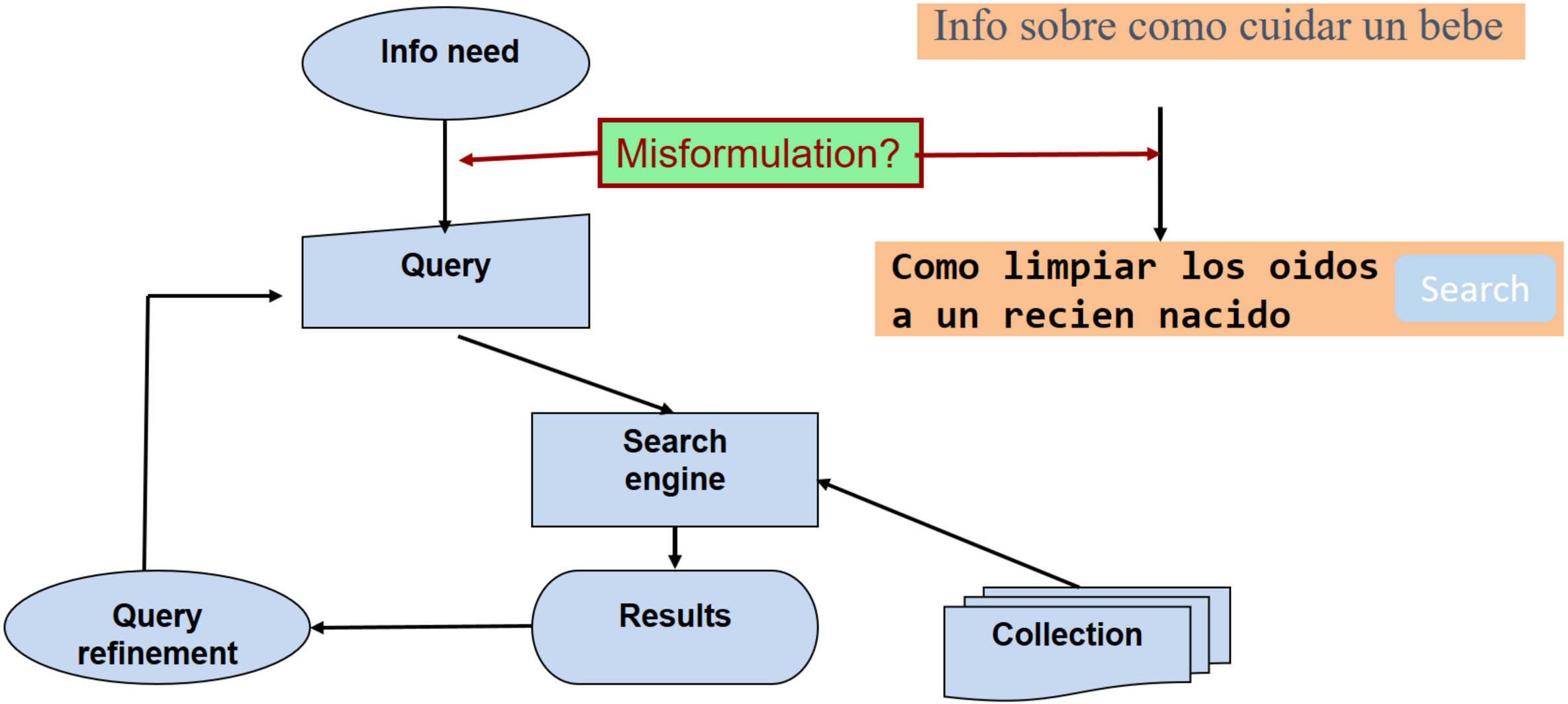
- Con frecuencia pensamos primero en la ***búsqueda web***, pero hay muchos otros casos:

  Búsqueda E-mail
  Búsquedas en sus computadores.
  Bibliotecas
  Herramientas: ElasticSearch, Apache Solr

Componentes del problema

- Colección: Un conjunto de documentos.
  Supongamos que es una colección estática por el momento.

- Objetivo: Recuperar documentos con información que sea relevante para la necesidad de información del usuario y ayude al usuario a completar una tarea.


Info need
Info sobre como cuidar un bebe
Misformulation?
Query
Como limpiar los oidos
a un recien nacido
Search
Search
engine
Query
refinement
Results
Collection

Operaciones de procesamiento de texto

Tokenización (I)

- ***Entrada:*** "Los amigos de Pedro"
- ***Salida:*** Tokens (unidad mínima para procesamiento)

  Los
  Amigos
  De
  Pedro

- ***Tokens***: es una instancia de secuencia de caracteres.
- Cada token es ahora un candidato para una entrada a la construcción de un ***índice***, después de un procesamiento adicional.
- Pero, ¿cuáles se consideran tokens válidos?

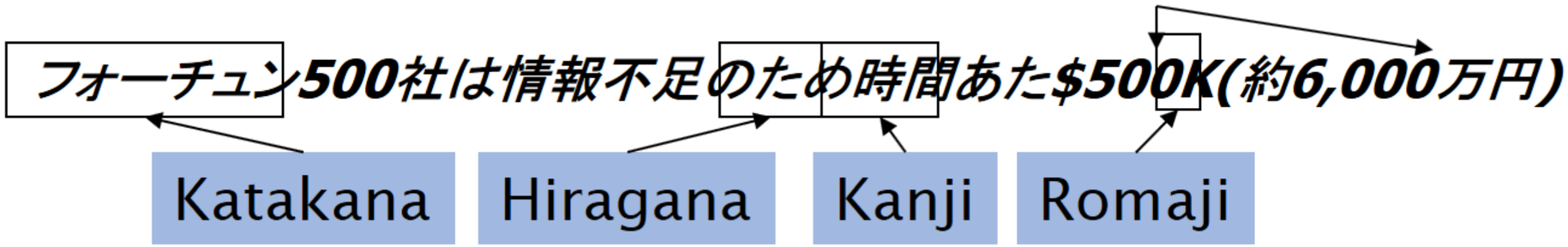
Tokenización (II)

- ***Retos***
  - Finland’s capital
  - Finland AND s? Finlands? Finland’s?
  - ***Hewlett-Packard*** → ***Hewlett*** y ***Packard*** como dos tokens?
    - ***state-of-the-art***: romper la secuencia de guiones?
    - ***co-educación***
  - **Música Ligera**
    - Un token o dos tokens?
  - **En alemán los sustantivos compuestos no se segmentan!!!**
    - Lebensversicherungsgesellschaftsangestellter
    - ‘life insurance company employee’
    - Los sistemas de recuperación alemanes se benefician enormemente de un módulo divisor compuesto. Puede dar un aumento de rendimiento del 15% para alemán.

Tokenización (III)

- ***Retos***
  - Chino no tiene espacio entre palabras
  - No siempre se garantiza una única tokenización
  - 莎拉波娃现在居住在美国东南部的佛罗里达。

  - Más complicado en japonés, con múltiples alfabetos entremezclados



Tokenización (IV)

- ***Números***
  - 3/20/91 Mar. 12, 1991 20/3/91
  - 55 B.C.
  - B-52
  - Password 324a3df234cb23e
  - (800) 234-2333

- Los sistemas IR más antiguos no indexaban números.
- Pero a menudo es muy útil: piense en cosas como buscar códigos de error/stacktraces en la web.

Palabras de parada (stop words)

- ***Con una lista de parada, se excluye del diccionario las palabras más comunes. Intuición:***
  - Tienen poco contenido semántico: la, a, y, de, como.
  - Hay muchos de ellos en todas colección. No sirve como criterio diferenciador de documentos.

- Pero la tendencia es no hacer esto:
  - Existen buenas técnicas de compresión, lo que significa que el espacio necesario para incluir palabras vacías en un sistema es muy pequeño.
  - Hacen parte de la sintaxis de una oración correcta son necesarias para frases como.
    - Rey de Dinamarca
    - Vuelos a Bogotá
  - Los recientes embeddings contextuales requieren de esta información.

DETERMINANTES

Acompañan al nombre, y lo determinan, concretando o limitando su extensión

EL ARTÍCULO	DEMOSTRATIVOS	INDEFINIDOS	POSESIVOS	NUMERALES	INTERROGATIVOS	EXCLAMATIVOS
el, las lo, los las, al, del.	este, esta, estos, estas, ese, esa, esos, esas, aquel, aquella, aquellos, aquellas.	un, uno, una, unos, unas, algún, alguno, alguna, algunos, algunas, ningún, ninguno, ninguna, ningunos, ningunas, poco, poca, pocos, pocas, escaso, escasa, escasos, escasas, mucho, mucha, muchos, muchas, demasiado, demasiada, demasiados, demasiadas, todo, toda, todos, todas, varios, varias, otro, otra, otros, otras, mismo, misma, mismos, mismas, tanto, tanta, tantos, tantas, cualquier, cualquiera, cualesquiera, tal, tales, bastante, bastantes.	mío, mi, tuyo, tu, suyo, su, nuestro, vuestro, suyo, su, míos, mis, tuyos, tus, suyos, sus, nuestros, vuestros, suyos, sus, mía, mi, tuya, tu, suya, su, nuestra, vuestra, suya, su, mías, mis, tuyas, tus, suyas, sus, nuestras, vuestras, suyas, sus.	uno, dos, tres, primero, segundo, tercero, mitad, doble, duplo, dúplice, tercio, triple, triplo, triplice, decimocuarto, catorceavo, cuadragésimo.	¿qué?, ¿cuánto?, ¿cuánta?, ¿cuándo?, ¿cuál?, ¿cuáles?, ¿dónde?, ¿quién?, ¿quiénes?	¡qué!, ¡cuánto!, ¡cuánta!, ¡cuándo!, ¡cuál!, ¡cuáles!, ¡dónde!, ¡quién!, ¡quiénes!

Cuando sustituyen al nombre al que refieren, cumplen la función de **pronombre.**

Normalización

- ***Es posible que necesitemos "normalizar" las palabras en el texto indexado y de la consulta.***

  - U.S.A y USA son el mismo.
  - Hay muchos de ellos en toda colección. No sirve como criterio diferenciador de documentos.

  - Acentos: p. ej., francés, español, currículum vs. curriculum.
  - Incluso en idiomas que normalmente tienen acentos, es posible que los usuarios no los escriban.

- El resultado son términos: un ***término*** es un tipo de palabra (normalizado), que es una entrada en nuestro ***diccionario*** del sistema IR.

- La tokenización y la normalización pueden depender del idioma y, por lo tanto, están entrelazadas con la detección del idioma.

Lematización

- ***Reducir las formas flexivas/variantes a la forma base***
  - **pan**: panadero - panadería - panecillo
  - pescar: pescado - pesquero - pescador – pescadería

  - Es mas fácil en ingles?:
  - **be**: am, are, is (verbal)
  - **car***: car, cars, car's, cars' (nominal)*

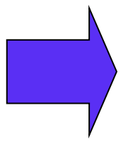
  - La lematización implica hacer una reducción "adecuada“.
    - Se requiere de diccionarios con la morfología de las palabras.

Stemming (I)

- ***Son reglas de corte (básicamente) que se construyen para cada lenguaje***
  - automatizar(s), automático, automatización todo reducido a automat.
  - automate(s), automatic, automation todo reducido a automat.

- Ejemplo de un algoritmo de stemming en ingles

for example compressed and compression are both accepted as equivalent to compress.



for exampl compress and compress ar both accept as equival to compress

Stemming (II)

- ***Para inglés se usa principalmente Porter stemmer en*** ***http://www.tartarus.org/~martin/PorterStemmer/***
  - Ejemplos:
    - ATIONAL -> ATE relational -> relate
    - TIONAL -> TION conditional -> condition
    - ENCI -> ENCE valenci -> valence
    - ANCI -> ANCE hesitanci -> hesitance
    - IZER -> IZE digitizer -> digitize
    - ABLI -> ABLE conformabli -> conformable
    - ALLI -> AL radicalli -> radical
    - ENTLI -> ENT differentli -> different
    - ELI -> E vileli -> vile
    - OUSLI -> OUS analogousli -> analogous
  - Se selecciona la regla con el sufijo mas largo.

Stemming (III)

- ***Stemming realmente sirve?***

  Inglés: resultados mixtos. Ayuda en recall a algunas consultas, pero perjudica la precisión en otras.

  - **Definitivamente útil para español, alemán, finlandés,…¡30 % de aumento de rendimiento para los finlandeses!**

Resumen pasos de procesamiento

- ***Tokenización***
- ***Normalización***
- ***Stemming***
- ***Palabras de paradas.***

- ***Luego de todos estos pasos:***
- Tenemos un conjunto de "tokens normalizados" que describen los documentos. Llamémoslos "***términos***".
- Nuestra hipótesis es que estos "***términos***" permiten discriminar documentos.
- ***Necesitamos una estructura para relacionar documentos con términos.***

Matrices termino-documento

Un ejemplo de búsqueda

- Que obras de Shakespeare contienen las palabras **Brutus** AND **Caesar** pero NOT **Calpurnia**.

- Se podría usar el comando GREP sobre todas las obras de Shakespeare's buscando **Brutus** AND **Caesar**, luego eliminar aquellas que contienen **Calpurnia**.

- ¿Por qué este enfoque no es la mejor opción?
  - Lento para grades corpus.
  - ***NOT Calpurnia*** es una búsqueda complicada
  - ¿Como renqueamos para poner los mas relevantes en primera posición?

Matriz Termino-Documento Binaria

- Cada documento es representado por un vector binario $\epsilon \{0,1\}^{|V|}$

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
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

1 si la obra contiene la palabra, 0 en cualquier otro caso.

Brutus *AND* ***Caesar***
BUT NOT ***Calpurnia***

Vectores de incidencia

- Básicamente tenemos un vector de 0/1 por cada termino.
- Para responder la consulta: Brutus AND Caesar AND NOT Calpurnia como opero esos vectores?
- Debo realizer un bitwise AND:
  - 110100 *AND*
  - 110111 *AND*
  - 101111 =
  - **100100**

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
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

Colecciones mas grandes

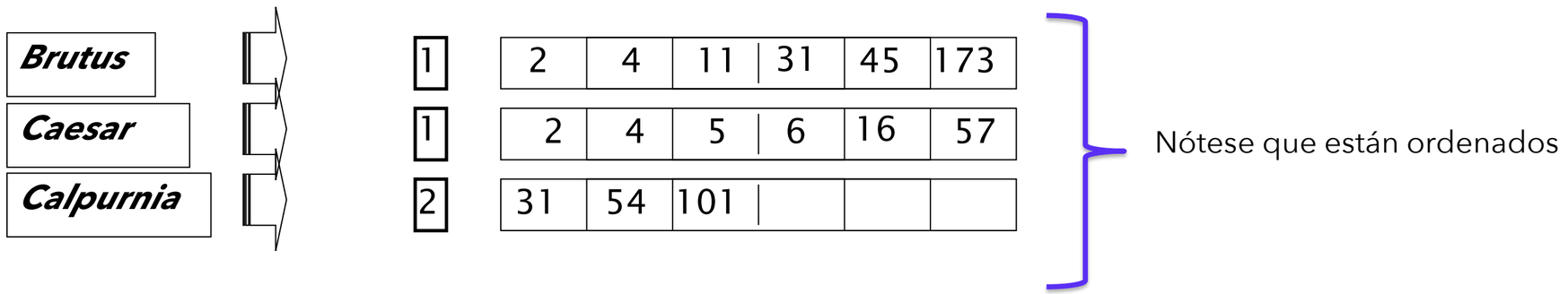
- Considere N=1 millón de documentos, cada uno con cerca de 1000 palabras.
- En promedio cada palabra se representa con 6 bytes/palabra incluyendo espacios/puntuación.
  - 6G de datos en los documentos.
- Suponga que existen alrededor de M=500k **términos** distintos en los documentos.
- 500K x 1M matriz que tiene 500.000.000.000 0's y 1's.
- Pero no tienen mas te un billón de 1's (matriz extremadamente dispersa)? **Por que?**
- ¿Que mejor representación se les ocurre?
  - Solo guardas las posiciones 1.

El índice invertido

Índice Invertido

- Para cada termino t, debemos almacenar una lista con los documentos que contienen t.
  - Identificar cada doc mediante un docID, un identificador único.

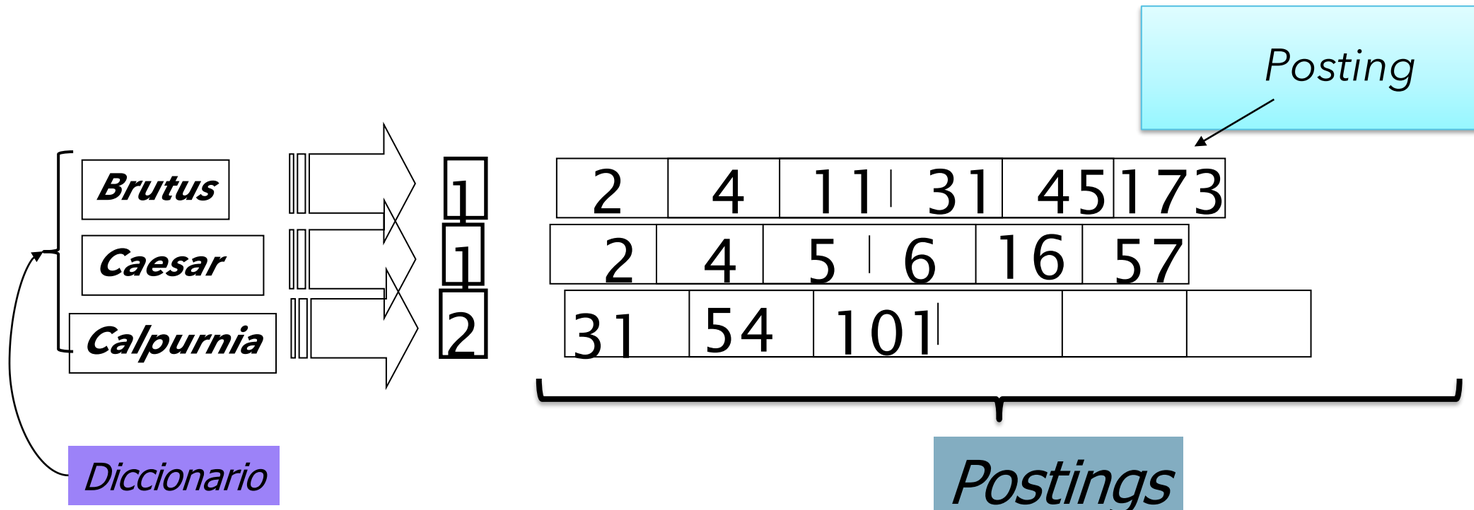
- ¿Podemos usar arreglos de tamaño fijo para esto?



¿Que sucede si la palabra Caesar se adiciona al documento 14?

Índice Invertido

- Es necesario una lista de tamaño variable para las listas de publicaciones (postings).
  - Generalmente se usan listas-vinculadas de longitud variable como estructura de datos.



Ordenado según docID

Construcción del Índice Invertido

Colección de documentos

Friends, Romans, countrymen.

Tokenizer

Secuencia de tuplas (token, docID)

Friends | Romans | Countrymen

Normalization + Stemming

Secuencia de tuplas (términos, docID)

friend | roman | countryman

Indexer

Indexación

friend	2 → 4 →
roman	1 → 2 →
countryman	13 → 16

Secuencia de Tokens

- Secuencia de pares: (tokens modificados (normalización), docID)

Documento 1

I did enact Julius Caesar I was killed i' the Capitol; Brutus killed me.

Documento 2

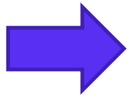
So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious

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

Secuencia de Tokens: Ordenar

- Ordenar por términos
  - Luego ordenar por docID
- Este paso es bastante importante.

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

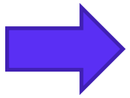


Term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

Indexar: Diccionario/postings

- Múltiples entradas de un mismo termino en un mismo documento se combinan.
- Dividir en un Diccionario y Postings.
- Finalmente agregar la información de la frecuencia a nivel de documento (i.e. cuantos documentos contienen el termino).
- ¿Como indexamos eficientemente?
- ¿Cuanto almacenamiento se requiere?

Term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1



term	doc. freq.	→	postings lists
ambitious	1	→	2
be	1	→	2
brutus	2	→	1 → 2
capitol	1	→	1
caesar	2	→	1 → 2
did	1	→	1
enact	1	→	1
hath	1	→	2
i	1	→	1
i'	1	→	1
it	1	→	2
julius	1	→	1
killed	1	→	1
let	1	→	2
me	1	→	1
noble	1	→	2
so	1	→	2
the	2	→	1 → 2
told	1	→	2
you	1	→	2
was	2	→	1 → 2

Como procesar una consulta usando el índice invertido

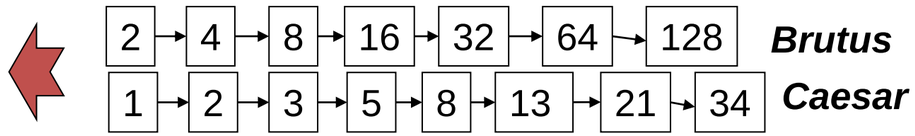
Consultas Booleanas: Match Exacto

- El modelo de recuperación booleano permite realizar consultas a través de operadores lógicos: AND, OR, NOT.
    - Se ve cada documento como un conjunto de palabras.
    - Es exacto: el documento cumple la condición o no la cumple.
    - El modelo mas simple de recuperación de la información.
- El modelo de recuperación booleano fue el motor de búsqueda comercial predilecto por mas de 3 décadas.
- Muchos sistemas aún lo utilizan (macOS Spotlight, Panamericana).

Procesamiento de consultas: AND

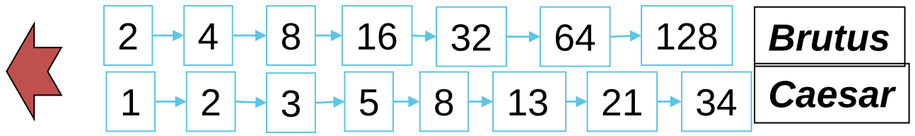
- Considere la siguiente consulta:
  - Brutus ***AND*** Caesar
- Que pasos debería seguir?
- Paso 1: Localice **Brutus** en el diccionario.
  - Recuperar la lista de postings.
- Paso 2: Localice **Caesar** en el diccionario.
  - Recuperar la lista de postings.
- Paso 3: “Mezclar” las dos listas de postings (operación de intersección)

Que algoritmo me permite
Realizar la intersección?



Operación de Mezcla (Algoritmo Merge)

- Recorra las dos listas de postings simultáneamente, en tiempo lineal en el número total de entradas.



- Si las longitudes de las listas son ***x*** y ***y***, la operación de mezcla toma ***O(x+y)***
- ***Se cumple siempre y cuando los postings estén ordenados.***

Operación de Mezcla (Algoritmo Merge)

```
INTERSECT(p1, p2)
 1  answer ← ⟨ ⟩
 2  while p1 ≠ NIL and p2 ≠ NIL
 3  do if docID(p1) = docID(p2)
 4        then ADD(answer, docID(p1))
 5             p1 ← next(p1)
 6             p2 ← next(p2)
 7        else if docID(p1) < docID(p2)
 8                then p1 ← next(p1)
 9                else p2 ← next(p2)
10  return answer
```

Recorra las dos publicaciones simultáneamente, en tiempo lineal en el número total de entradas de publicaciones

Consultas Booleanas (Otras)

- Ejercicio: Adapte el algoritmo de mezcla para los siguientes queries.
  - Brutus ***AND NOT*** Caesar
  - Brutus ***OR NOT*** Caesar
- La operación de mezcla aún puede correr en tiempo O(x+y)?, Que podemos lograr a nivel de complejidad?

- Brutus **AND NOT** Caesar **O(x+y)**
- Brutus **OR NOT** Caesar **O(N)**

Recorra las dos publicaciones simultáneamente, en tiempo lineal en el número total de entradas de publicaciones

Consultas Booleanas (Complejidad General)

- Suponga cualquier combinación de query, por ejemplo:
  - ***(Brutus OR Caesar) AND NOT (Antony OR Cleopatra)***

- ¿Cual seria la complejidad expresado como una cota superior?

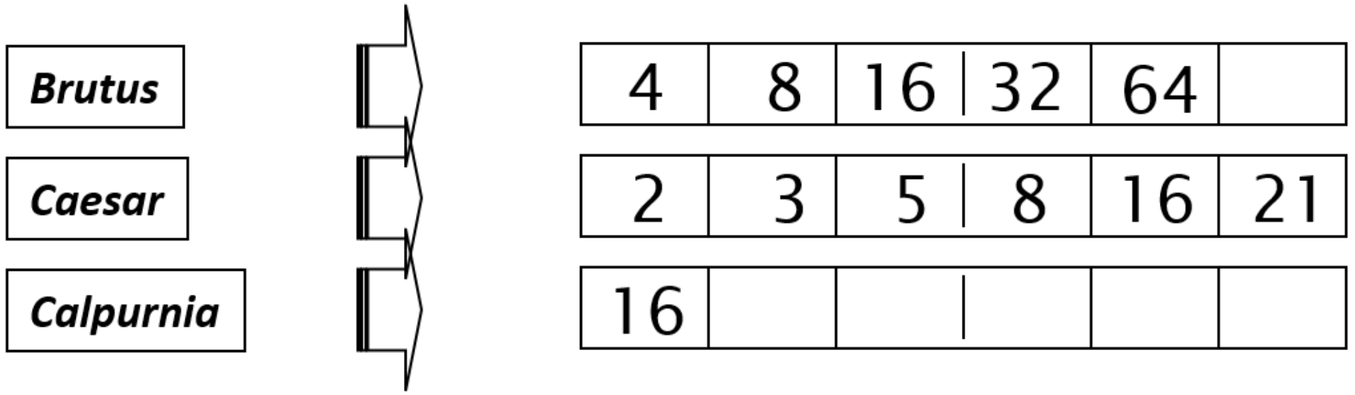
- **O(qN)** donde q es el número de términos.

  - ¿Podemos optimizar la consulta?

Recorra las dos publicaciones simultáneamente, en tiempo lineal en el número total de entradas de publicaciones

Optimización de Consultas (I)

- Cual es el mejor orden para el procesamiento de consultas?
- Consideremos el caso mas fácil una consulta conjuntiva (solo ANDs).
- **Por cada termino obtener sus postings y luego realizar el AND**
  - ***En que orden?***



Consulta: ***Brutus*** *AND* ***Calpurnia*** *AND* ***Caesar***

- Procesar en orden incremental de acuerdo a la frecuencia.

Recorra las dos publicaciones simultáneamente, en tiempo lineal en el número total de entradas de publicaciones

Optimización de Consultas (II)

- Recomiéndenme un orden para el procesamiento del siguiente query:

***(tangerine** OR **trees)** AND*
***(marmalade** OR **skies)** AND*
(kaleidoscope** OR **eyes)

¿Cual par de términos deberíamos procesar primero?

Término	Doc.Freq
eyes	213312
kaleidoscope	87009
marmalade	107913
skies	271658
tangerine	46653
trees	316812

Recorra las dos publicaciones simultáneamente, en tiempo lineal en el número total de entradas de publicaciones

Optimización de Consultas (III)

- Procedimiento general:
  - Obtenga la frecuencia de todos los términos.
  - Estime el tamaño de cada subconjunto OR como la suma de sus frecuencias.
  - Procese en orden incremental.

Recorra las dos publicaciones simultáneamente, en tiempo lineal en el número total de entradas de publicaciones

Limitaciones

- Bueno para usuarios expertos con una comprensión precisa de sus necesidades y la colección.
- No es bueno para la mayoría de los usuarios.
  - La mayoría de los usuarios son incapaces de escribir consultas booleanas (o lo son, pero creen que es demasiado trabajo).
- La mayoría de los usuarios no quieren leer miles de resultados.
  - Esto es particularmente cierto para búsqueda en la Web.
- Las consultas booleanas suelen generar muy pocos (=0) o demasiados (miles) de resultados.
  - Query 1: "*procesamiento de lenguaje natural*" → 200,000 resultados
  - Query 2: "*procesamiento de lenguaje natural, lematizador para arameo*" → 0 resultados.
- Se necesita mucha habilidad para generar una consulta que produzca una cantidad manejable de resultados.

Referencias

- Introduction to information retrieval (Chapter 6) https://nlp.stanford.edu/IR-book/
- Jurafsky D. and Martin J. (2021) Speech and Language Processing (3rd ed. draft). Online: https://web.stanford.edu/~jurafsky/slp3/
- Yoav Goldberg (2017). Neural Network Methods in Natural Language Processing.
- In Deng, L., & In Liu, Y. (2018). Deep learning in natural language processing.