# Recuperación de Información Multimedia

## Descriptores de Texto (BoW, Tf-Idf, LSA)

**CC5213 – Recuperación de Información Multimedia**
Departamento de Ciencias de la Computación
Universidad de Chile
Juan Manuel Barrios – https://juan.cl/mir/ – 2020

# Modelos Estadísticos de Texto

- El lenguaje es una construcción compleja, depende del entorno y cambia en el tiempo
- El problema de codificar las reglas de gramática que definen el lenguaje es demasiado difícil para lograr buenos resultados
- En análisis de texto, usar modelos estadísticos simples sobre grandes cantidades de datos usualmente obtiene mejores resultados que implementar reglas complejas a mano considerando pocos datos

Ver Halevy, Norvig, Pereira, The unreasonable effectiveness of data. 2009.
https://research.google.com/pubs/archive/35179.pdf

# Modelo Bag-of-Words

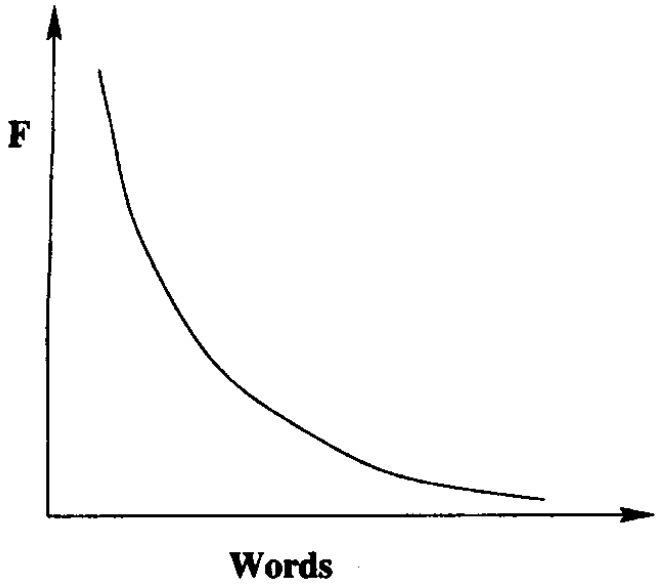
- Un modelo se refiere al enfoque usado para representar documentos y calcular relevancias
- El modelo Bag-of-Words (BoW) representa documento mediante estadísticas de sus términos
- En principio, los términos corresponden a las palabras aisladas (se pierde la relación entre palabras)
- Requiere conocer el universo de términos posibles (vocabulario)

# Frecuencias de Palabras

- **Ley de Zipf**: la frecuencia de la *i*-ésima palabra más frecuente es:

$$f_i = f_1 \cdot \frac{1}{i^\alpha}$$

  - $f_1$ es la frecuencia de la palabra más frecuente
  - El valor de $\alpha$ depende del texto (entre 1.5 y 2.0)
  - Es un tipo de “ley de potencia”




Ver Baeza 6.5

# Descriptor de Texto

- Sea D={$d_1$, …, $d_n$} el conjunto de todos los documentos en el dataset
- Sea K={$k_1$, …, $k_t$} el conjunto de todos los términos distintos en el dataset (vocabulario)
- Para cada término $k_i$ y documento $d_j$, se calcula un peso $w_{ij}$ con la relevancia de $k_i$ para $d_j$
  - Todos los $w_{ij} \geq 0$ y $w_{ij}=0$ ssi $k_i$ no existe en $d_j$
  - Se define la función que calcula pesos $g_i(d_j)=w_{ij}$
- A cada documento se le asocia un vector:

$$d_j = (w_{1j}, ..., w_{tj})$$

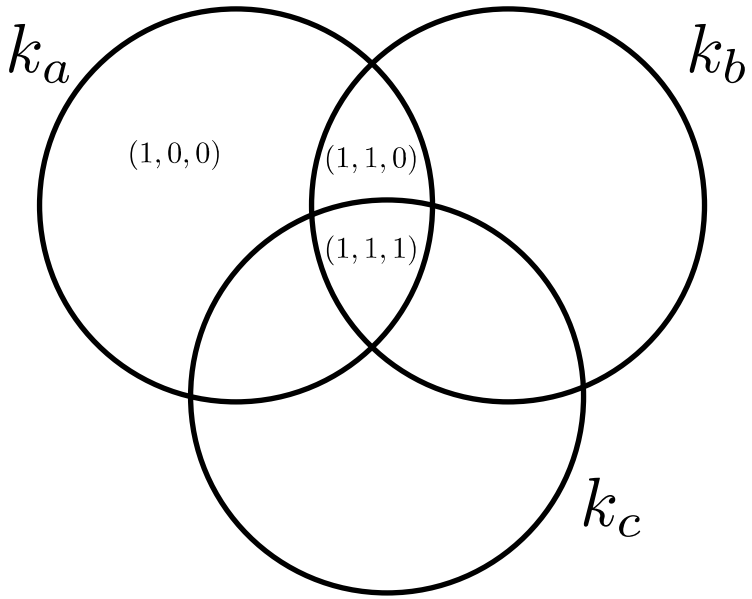
# Modelo Booleano

# Modelo Booleano

- Modelo booleano considera que los términos están presentes o ausentes en un documento
  - Cada $w_{ij}$ en {0, 1}
- Consultas se especifican como expresiones booleanas
  - Se consulta por la ocurrencia o no de cada término
  - Se pueden utilizar conectores lógicos: *not*, *and*, *or*
- Criterio de decisión binario, un documento es relevante o no lo es

# Ejemplo

- Supongamos existen tres términos: $K = \{k_a, k_b, k_c\}$
- Se realiza la consulta: $q = k_a \wedge (k_b \vee \neg k_c)$
- La consulta se convierte a la forma normal disjunta (DNF): $q_{dnf} = (1,1,1) \vee (1,1,0) \vee (1,0,0)$




Ver Baeza 3.2.2

# Consultas

- La consulta se resuelve comparando cada cláusula $q_{cc}$ de $q_{dnf}$ con cada documento
- Un documento es relevante si existe una cláusula de la consulta que coincidan todos sus pesos con los del documento

$$sim(q, d_j) = 1 \Leftrightarrow \exists q_{cc} \,|\, \forall i,\, w_i(d_j) = w_i(q_{cc})$$

Ver Baeza 3.2.2

# Modelo Booleano

- Permite obtener todos los documentos que cumplen cierta condición
  - Bases de datos, Recuperación de datos
- Ventajas:
  - Formalismo claro y preciso
  - Simple
- Desventajas:
  - No existe un grado de similitud o relevancia, es decir, $sim(q,d_j)$ es 1 o 0
  - La búsqueda usualmente recupera muy pocos o demasiados documentos

# Modelo Vectorial

# Modelo Vectorial

- Asignar pesos no binarios a los términos
  - Estos pesos se utilizan para determinar el grado de similitud entre documentos
- El modelo vectorial también toma en consideración documentos que sólo tienen un match parcial con la consulta
- Ranking: los documentos se priorizan de acuerdo a su grado de similitud con la consulta

# TF: Term Frequency

- Representar las características que permiten agrupar documentos similares (similitud intra-cluster)
  - Representar los términos que aparecen en el documento
- Sea $freq_{ij}$ la frecuencia del término $k_i$ en el documento $d_j$, el factor *tf* está dado por:

$$tf_{ij} = \begin{cases} 1 + \log(freq_{ij}) & \text{si } freq_{ij} > 0 \\ 0 & \text{si no} \end{cases}$$

# IDF: Inverse Document Frequency

- Representar las características que permiten distinguir documentos que no son similares (disimilitud inter-cluster)
  - Términos que aparecen en muchos documentos no son útiles para distinguir documentos
- Sea $N$ el número de documentos, $n_i$ el número de documentos donde aparece el término $k_i$. La frecuencia de documento inversa para $k_i$ es:

$$idf_i = \log\left(\frac{N}{n_i}\right)$$

# Descriptor TF-IDF

- Un documento $d_j$ se representa por un vector de $t$ dimensiones:

$$d_j = (w_{1j}, ..., w_{tj})$$

  - Donde $w_{ij}$ es el peso del término $i$ para el documento $j$, se calcula como:

$$w_{ij} = tf_{ij} \times idf_i$$

$$w_{ij} = \begin{cases} (1 + \log(freq_{ij})) \times \log(\frac{N}{n_i}) & \text{si } freq_{ij} > 0 \\ 0 & \text{si no} \end{cases}$$

# Largo de Documentos

- La cantidad de palabras en un documento influye en el término tf
- Normalizar cada vector de pesos a largo unitario
  - Los vectores se ubicarán en la superficie de una hiper-esfera de radio 1

$$d'_j = (w_{1j}, ..., w_{tj})$$

$$||d'_j|| = \sqrt{\sum_{i=1}^{t} w_{ij}^2}$$

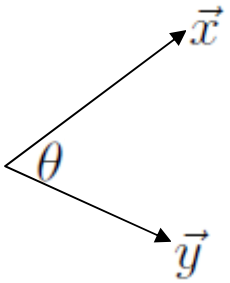
$$d_j = \frac{d'_j}{||d'_j||}$$

# Función de Similitud

- El modelo vectorial propone una forma para medir el grado de similitud entre documentos
- Una función de similitud comúnmente usada es el coseno del ángulo entre vectores:

Producto escalar, producto interno
o producto punto:

$$\vec{x} \cdot \vec{y} = ||\vec{x}|| \; ||\vec{y}|| \; \cos(\theta)$$

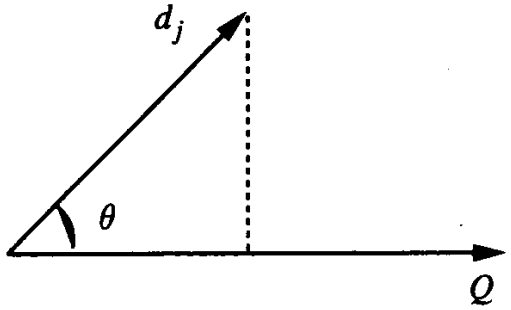


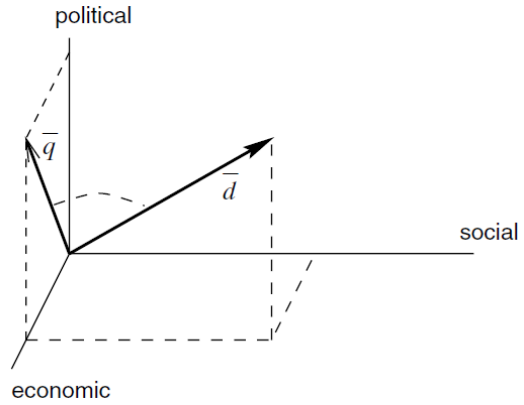

$$\cos(\theta) = \frac{\vec{x} \cdot \vec{y}}{||\vec{x}|| \; ||\vec{y}||}$$

$$\cos(\theta) = \frac{\sum_{i=1}^{t} x_i \, y_i}{\sqrt{\sum_{i=1}^{t} x_i^2} \; \sqrt{\sum_{i=1}^{t} y_i^2}}$$

# Similitud Coseno

- La similitud coseno mide el parecido de dos documentos:




$$q = (w_{1q}, ..., w_{tq})$$
$$d_j = (w_{1j}, ..., w_{tj})$$

$$sim(q, d_j) = \cos(\theta) = \sum_{i=1}^{t} w_{iq}\, w_{ij}$$

Ver Baeza 3.2.6

# Función de Disimilitud

- El coseno del ángulo mide similitud:
  - 1 → vectores idénticos
  - 0 → sin términos comunes
- Es posible convertir la similitud en disimilitud:

$$dis(q, d_j) = 1 - sim(q, d_j)$$

  - 0 → vectores idénticos
  - 1 → sin términos comunes
  - Sin embargo no es una función de distancia apropiada porque no cumple la desigualdad triangular

# Métricas

- Se puede obtener una métrica usando el ángulo (distancia angular):

$$dis(q, d_j) = 1 - \frac{\arccos(sim(q, d_j))}{\pi}$$

- Distancia euclidiana en función de similitud coseno:

$$dis(q, d_j) = \sqrt{2(1 - sim(q, d_j))}$$

$$\begin{aligned} ||\vec{a} - \vec{b}||^2 &= \sum_{i=1}^{n}(a_i - b_i)^2 \\ &= \sum_{i=1}^{n} a_i^2 + \sum_{i=1}^{n} b_i^2 - 2\sum_{i=1}^{n} a_i b_i \\ &= 1 + 1 - 2\cos\theta \end{aligned}$$

# Modelo Vectorial

- Ventajas del modelo vectorial
  - El esquema de pesos mejora la eficacia
  - Se pueden recuperar documentos que se aproximen a la consulta
  - La medida de similitud permite rankear documentos
- Desventaja
  - Se asume que los términos son independientes

# Generalización de Términos

# Tokenizer

- Es la componente que divide la cadena de caracteres del documento original en sus términos o palabras
- Debe decidir qué caracteres componen una palabra y cuales son separadores:
  - ¿Se diferencian mayúsculas, minúsculas y acentos?
  - ¿Son símbolos como @ ; : _ - ’, parte de una palabra?
  - ¿Considerar números y fechas? ¿reemplazarlos por símbolos?
- Por ejemplo, una opción simple es que el Tokenizer use una expresión regular como: `([A-Za-z0-9_]+)`

```
"El Sol salió a las 07:30"
  → [El, Sol, salió, a, las, 07, 30]
  → [el, sol, salio, a, las, #NUMERO#, #NUMERO#]
```

# Stop Words

- Se llaman “Stop Words” a las palabras sin significado (conectores, preposiciones, artículos) que es posible no considerar al analizar el contenido de un documento
  - Usualmente son las palabras más frecuentes
- Se debe notar que:
  - La lista de stop words (stop-list) depende del idioma
  - Un término que aparece en todos los documentos no es útil para definir relevancia (sea o no una stop word)
  - Hay casos donde sí interesan todas las palabras

# Stemming

- Considerar como término el inicio de cada palabra (word stem) que es compartido por todas sus variantes e inflexiones
  - No confundir con la raíz (root word) que se refiere al trozo relevante para su semántica (sin prefijos ni sufijos)
- Algoritmo de Porter es un método simple y muy usado
  - No produce necesariamente un resultado correcto (en el sentido lingüístico)

| | | | | | |
|---|---|---|---|---|---|
| ATIONAL | → | ATE | Relational | → | Relate |
| TIONAL | → | TION | Conditional | → | Condition |
| ENCI | → | ENCE | Valenci | → | Valence |
| ANCI | → | ANCE | Hesitanci | → | Hesitance |
| IZER | → | IZE | Digitizer | → | Digitize |

https://tartarus.org/martin/PorterStemmer/

# Lemmatization

- Convertir cada palabra en su palabra base (lema) que corresponde a la palabra que aparece en un diccionario.
- El lema depende del significado de la palabra, el cual depende del contexto
  - La palabra “traje” puede referirse al verbo (lema “traer”) o al sustantivo (lema “traje”). Se necesita el contexto de uso para decidir.
- Lemmatization puede llegar a ser bastante complejo en su implementación
  - Usualmente logra una mejora muy leve en efectividad con respecto a una técnica simple como stemming

# WordNet

- Base de datos de palabras en inglés.
- Cuatro tipos de palabras: adjetivos, adverbios, sustantivos y verbos.
- Define grupos de palabras con igual significado (synsets)
- Cada synset tiene un código y una definición
- Define relaciones entre synsets
  - Mayor/Menor, Tiene/Es_parte
- Permite calcular distancia semántica entre palabras

https://wordnet.princeton.edu/

# N-Grams

- Considerar como términos en un documento las secuencias de n palabras consecutivas
  - En vez de palabras pueden ser secuencias de lemmas, stems, caracteres, etc.
- El tamaño del vocabulario (términos en los documentos) aumenta en forma significativa
  - Solo usar los n-grams que superan una cantidad mínima de repeticiones
- Modelo “bag-of-words” se convierte en un modelo “bag-of-n-grams”
- Notar que para usar n-grams o stems basta modificar el Tokenizer

# N-Grams de Palabras

- Secuencias de *n* palabras consecutivas como un sólo término
  - También se incluyen secuencias de grado menor
- Por ejemplo:

| el | perro | feroz | juega | con | la | pelota | roja |
|---|---|---|---|---|---|---|---|

**8 unigramas**: el, perro, feroz, juega, con, la, pelota, roja

**7(+8) bigramas**: el_perro, perro_feroz, feroz_juega, juega_con, con_la, …

**6(+15) trigramas**: el_perro_feroz, perro_feroz_juega, feroz_juega_con, …

**5(+21) 4-gramas**: el_perro_feroz_juega, perro_feroz_juega_con, …

# N-Grams de Caracteres

- Secuencias de *n* caracteres consecutivos
  - También incluir las secuencias de grado menor
- Naturalmente consideran word stems
- Por ejemplo:

| e | l | | p | e | r | r | o | | y | | e | l | | p | e | r | r | i | t | o | | j | u | e | g | a | n |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

**char-unigramas**: e, l, _, p, e, r, r, o, _, y, _, e, l, _, p, e, r, r, i, t, o, _, j, u, …

**char-bigramas**: e-l, l- , -p, p-e, e-r, r-r, r-o, o- , -y, y- , e-l, …

**char-trigramas**: e-l- , l- -p, -p-e, p-e-r, e-r-r, r-r-o, r-o- , o- -y, -y- , …

**char-4-gramas**: e-l- -p, l- -p-e, -p-e-r, p-e-r-r, e-r-r-o, r-r-o- , r-o- -y, …

# Ejercicio

- Se tiene un dataset de 1.000 documentos con las siguientes estadísticas del vocabulario:

| Palabra | Cantidad de veces que aparece en el dataset | Número de documentos en los que aparece |
|---|---|---|
| clavito | 1.000 | 10 |
| clavó | 10 | 10 |
| Pablito | 10.000 | 1.000 |
| qué | 100 | 100 |
| un | 1.000 | 100 |

- Usando TF-IDF calcular la similitud entre las frases:
  - Documento 1: “Pablito clavó un clavito”
  - Documento 2: “¿Qué clavito clavó Pablito?”

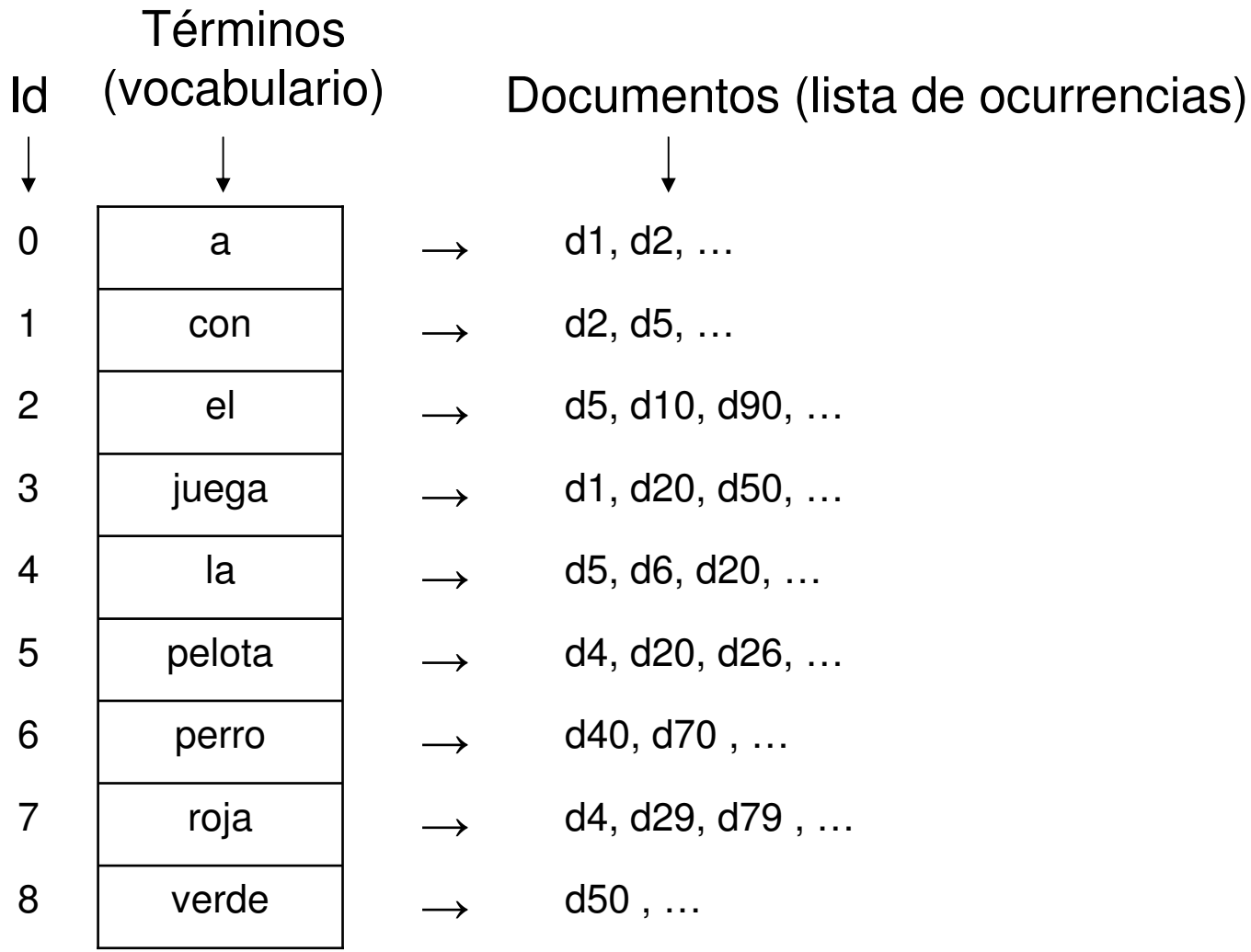
# Búsqueda

# Índice Invertido

- El vocabulario contiene todos los términos en un dataset
- Las lista de ocurrencias de un término son todas las ubicaciones donde aparece en el dataset
  - Cada ocurrencia contiene (al menos) el id de documento
  - Dependiendo de la aplicación, se puede guardar más información como: posición absoluta dentro del documento (indexOf), número de palabra, número de página, número de párrafo, etc.
- El índice invertido es una tabla que asocia cada término con su lista de ocurrencias

Ver Baeza, Cap.8.2

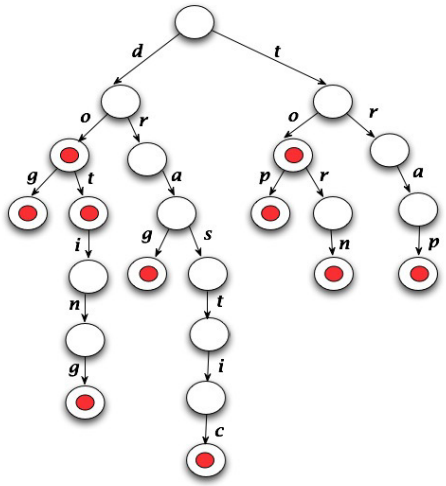
# Índice Invertido



# Búsqueda en el Vocabulario

- Para calcular cada descriptor es necesario un método eficiente para determinar el id en el vocabulario de cada término
  - Array ordenado de todo el vocabulario y usar búsqueda binaria: O(log(n))
  - Árbol de búsqueda
    - Trie=Árbol de prefijos: O(len(s))
    - Radix Tree o Patricia Tree

    comprimen caminos



# Distancia entre palabras

- Distancia de Levenshtein (o distancia de edición)
  - *d(s,t)* es el número mínimo de operaciones para transformar el string *s* en el string *t*
  - Cada operación de borrar, agregar o reemplazar un carácter tiene un costo fijo 1
  - Costo evaluación: *O(n m)*
  - Cumple las propiedades métricas
- Variantes:
  - Costo de inserción es mayor que costo de reemplazo
  - Costo de reemplazo variable: $c(n \rightarrow ñ) < c(n \rightarrow x)$
  - Operación de transposición: ab→ba

# Requerimientos de Espacio

- El vocabulario usualmente requiere poco espacio
  - Se asume que es posible mantener el vocabulario en memoria
- Las listas de ocurrencias usualmente requieren mucho espacio
  - Almacenado contiguo de todas las ocurrencias (posting file)
  - Usar almacenamiento secundario y recuperar listas por palabra desde el disco
  - Algoritmos de compresión y búsqueda en texto comprimido

# Búsqueda Eficiente

- La consulta se tokeniza, se obtienen sus términos y se calcula su vector tf-idf
- Con el índice invertido se obtienen las listas de ocurrencias que contienen los términos en la consulta
- Intersectar las listas de ocurrencias (seleccionar documentos con todos o casi todos los términos)
  - Para intersección eficiente las listas de ocurrencias deben estar ordenadas
  - Se puede realizar un merge lineal, o si una lista es mucho más corta (caso probable por la ley de Zipf) usar búsqueda binaria
- Los documentos seleccionados se ordenan según su similitud con la consulta (i.e., sim.coseno)

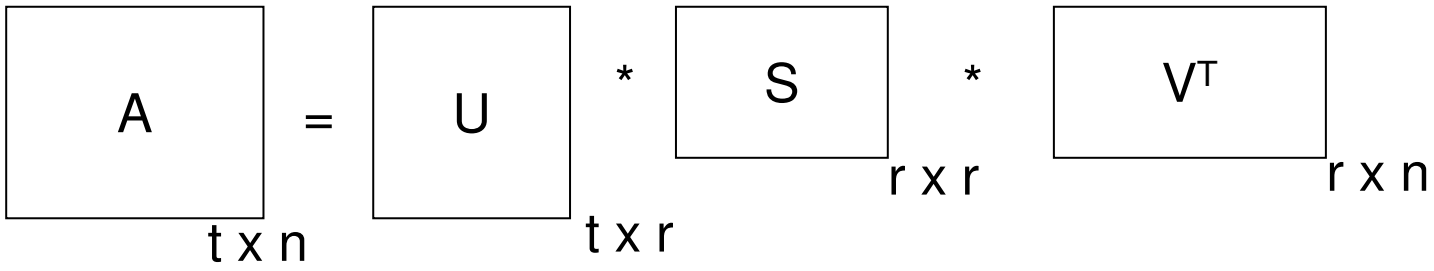
# Latent Semantic Analysis

# Latent Semantic Analysis

- **Latent Semantic Analysis** (LSA) o **Latent Semantic Indexing** (LSI)
- Para un vocabulario de *t* términos y un dataset de *n* documentos, sea *A* la matriz de descriptores:
  - *A* una matriz de *t*x*n* con vectores (ej: tf o tf-idf) como columnas
  - $A^TA$ es una matriz de *n*x*n* contiene la similitud de documentos según términos comunes
  - $AA^T$ es una matriz de *t*x*t* contiene la similitud de términos según documentos comunes
- Idea: Eliminar términos correlacionados para reducir la cantidad de términos necesarios para representar cada documento del dataset
- Parecido a reducir dimensiones con PCA aunque operando directamente sobre A

# Latent Semantic Analysis

- Usando el método Singular Value Decomposition se factoriza la matriz $A=USV^T$
  - U matriz de *t* x *r*
    - Contiene los vectores propios de $AA^T$
  - S matriz diagonal de *r* x *r*
    - Contiene la raíz cuadrada de los valores propios de $AA^T$
    - Los *r* valores singulares $\lambda_1,\ldots,\lambda_r$ son distintos de cero
    - $r \leq \min(t,n)$ se llama el rank de A
  - V matriz de *r* x *n*
    - Contiene los vectores propios de $A^TA$



# Latent Semantic Analysis

- U y V son ortonormales:
  - $U^T U = I$
  - $V^T V = I$
- Se demuestra que:
  - $AA^T = (USV)(USV)^T = (USV)(V^T S^T U^T) = US^2U^T$
  - $A^T A = (USV)^T(USV) = (V^T S^T U^T)(USV) = V^T S^2 V$

# Latent Semantic Analysis

- Seleccionar los *k* valores singulares de mayor magnitud y reducir la dimensión de las matrices de *r* a *k*
  - $A' = U' S' V'^{T}$
  - U' de *t* x *k* representa el espacio de los conceptos
  - $S'V'^{T}$ de *k* x *n* son los documentos en ese espacio

# Latent Semantic Analysis

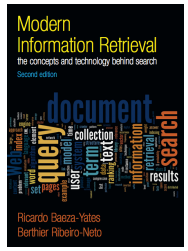
- Convertir los descriptores de documentos a su representación "**latent semantic**"
  - Usar $V'^{T}$ o alternativamente multiplicar $A^{T}$ por $U'S'^{-1}$
- Para una consulta:
  - Calcular su descriptor q original (tf o tf-idf) de dimensión t
  - Obtener una representación de q que pueda ser comparada con los documentos
    - $q' = q^{T}U'S'^{-1}$
  - Comparar q' con cada documento usando similitud coseno

# Latent Semantic Analysis

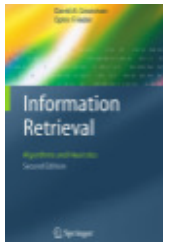
- Se reduce las dimensiones del vector, agrupando palabras similares
- Similitud entre palabras se refiere a palabras que tienden a aparecer simultáneamente
- A las nuevas dimensiones se les llama el espacio "**semántico latente**"
- Los documentos se representan por conceptos semánticos en vez de términos o palabras
- Lento de calcular (operaciones entre matrices grandes)

# Bibliografía

- **Modern Information Retrieval**. Baeza-Yates, Ribeiro-Neto, 2011.
  - Cap. 3, 6, 8 y 9
- **Multimedia Retrieval**. Blanken, de Vries, Blok, Feng. 2007.
  - Cap. 3.5, 4.3 y 4.4.
- **Information Retrieval. Algorithms and Heuristics. Second Edition.** Grossman, Frieder. 2004
  - Cap 2.6 (LSA)







# Herramientas útiles

- Python, scikit-learn
  - http://scikit-learn.org/stable/modules/feature_extraction.html
- Java, Lucene
  - https://lucene.apache.org/core/
- R, package tm
  - http://tm.r-forge.r-project.org/