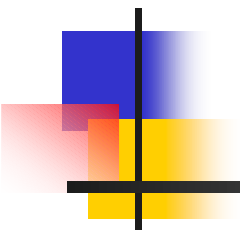


Análisis de los documentos

ARI – Curso 2001/2002

Características del lenguaje

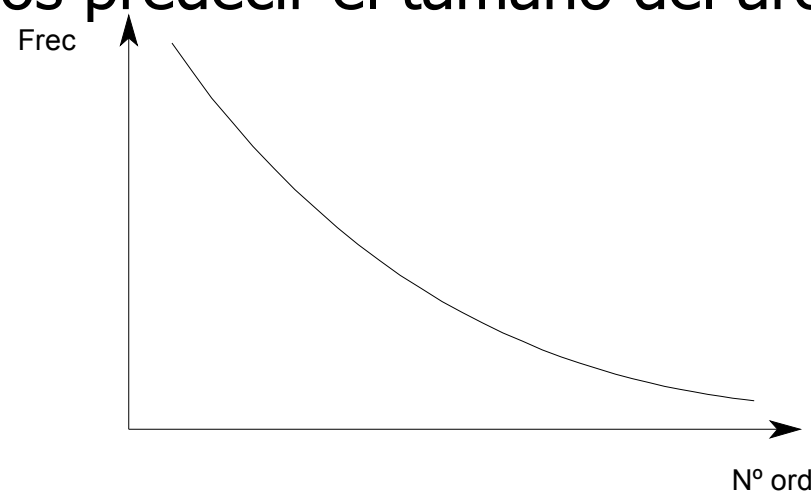
- Ley de Zips
  - En los 40's, Zips analizó la relación entre la frecuencia y la palabra
  - Las frecuencias de las palabras en un lenguaje natural no son equiprobables
  - Ley de Zips:

$$n * f(n) \approx cte$$

    - El orden de una palabra X su frecuencia es cte
    - Ordenando las palabras de mayor frecuencia de aparición a menor

Características del lenguaje

- Ley de Zips
  - Las estadísticas del corpus de Brown (1967) la corroboran
  - Podemos predecir la frecuencia de una palabra según su orden
  - Podemos predecir el tamaño del archivo invertido



Características del lenguaje

- Ley de Heaps
  - El tamaño del corpus (≡ Nº de palabras de la colección) está relacionado con el tamaño del diccionario

$$t = k * N^{\beta}$$

N = Nº de palabras de la colección

t = Nº de palabras distintas (≡ tamaño del diccionario)

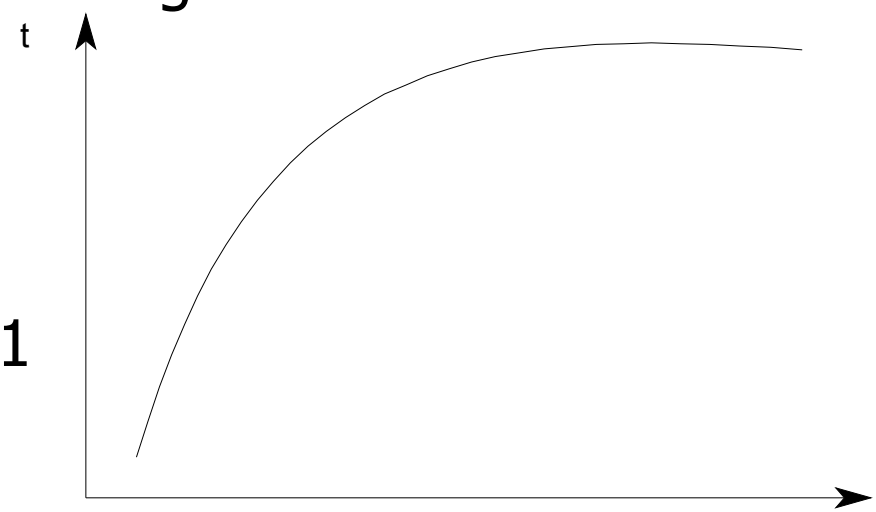
Características del lenguaje

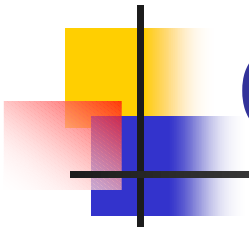
- Ley de Heaps
  - A medida que aumenta N, el diccionario crece
  - K y β dependen del idioma y de la colección
  - Aproximadamente en inglés

    $10 \leq k \leq 20$

    $0'5 \leq \beta \leq 0'6$

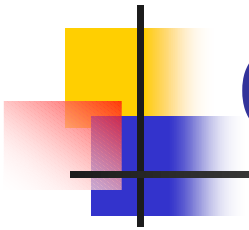
  - β no puede ser > 1





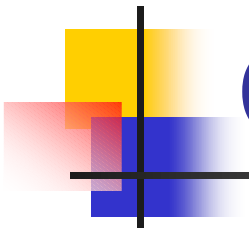
Características del lenguaje

- Ley de Heaps
  - Podemos predecir el tamaño del diccionario
  - Si el corpus es muy grande el tamaño del diccionario será casi fijo
    - Importante si la implementación es una tabla Hash



Clases de documentos

- Sin estructura:
  - Texto libre
  - Secuencia de palabras
  - Utilizamos una indexación de por términos
  - Realizamos las consultas como ya hemos visto

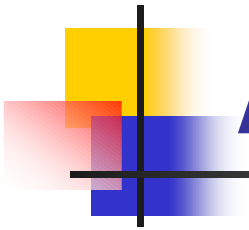


Clases de documentos

- Estructura fija
  - El documento está estructurado en campos
  - La estructura es externa al documento
  - La estructura la conocen los programas que manipulan el documento
  - Es interesante contemplar la estructura para indexar
    - BD Relacional
    - BDR + Indexación de texto libre
  - Ej: e-mail o una ficha bibliográfica

Clases de documentos

- Documento con Metadatos
  - El documento está estructurado en campos
  - La estructura es interna al documento
  - Ej: XML
    - DTD: Estructura del doc = Metadatos
    - XML: Datos
  - Debemos tener en cuenta la estructura para indexar



Aspectos de indexación

- Indexar: Obtener los términos que modelizan el contenido de un doc
- Hipótesis: Las palabras del texto son candidatos a términos
- El problema es identificar las palabras y los términos

Aspectos de indexación

- Dígitos
  - Generalmente son palabras poco específicas
  - Hay casos en que si son importantes
    - Vitamina B12
    - Efecto 2000
  - No hay una solución general pues depende del dominio
  - En colecciones generales se consideran los que empiezan por letra
    - 2000 → NO
    - B12 → SI

Aspectos de indexación

- Guiones
  - Al final de palabra: Se puede eliminar y juntar los dos trozo
  - Separador:
    - Es bueno que siga junto: F-16
    - Son dos palabras: Jean-Claude
  - En un dominio específico tenemos otros elementos de juicio

Aspectos de indexación

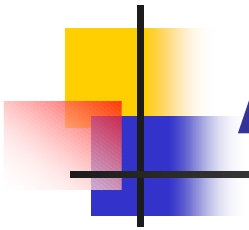
- Guiones
  - En dominios generales podemos usar:
    - F no es término y 16 no es término
      Por tanto F-16 es término
    - Jean es término y Claude es término
      Por tanto "–" es un separador

Aspectos de indexación

- Otros signos
  - Puntuación, barras,...
  - En la mayoría de los casos son separadores
  - Aunque no siempre
    - Ej: OS/2
- Mayúsculas/Minúsculas
  - Normalmente da lo mismo
  - Se suelen poner todo en mayúsculas o todo en minúsculas

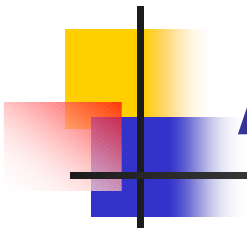
Aspectos de indexación

- Palabra
  - Regla general (colección genérica):
    - Palabra: Toda cadena de caracteres alfanuméricos o numéricos en que el primer carácter es una letra. Todos los caracteres se pasan a mayúsculas (o minúsculas) y todos los demás son separadores.
  - En dominios específicos se tiene una lista de términos candidatos
    - Ej: B12 no lo separes



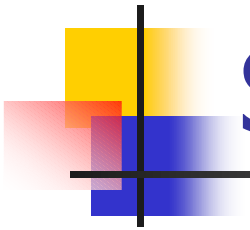
Aspectos de indexación

- En las preguntas se suele hacer el mismo análisis léxico
  - Ej: “Quiero documentos que tengan 13 **animales** de **pelo verde** y **rojo**”



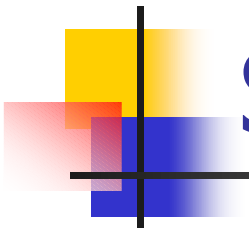
Analizador léxico

- Diseño:
  - Algoritmo específico para nuestro dominio
  - Maquina de estados (Autómata finito)
    - Técnica clásica
  - Generador de analizadores léxicos
    - LEX
    - Pueden tener problemas para tratar errores



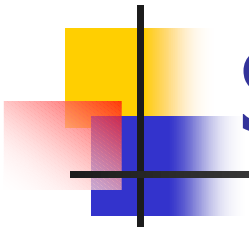
STOPLIST

- En un primer paso todas las palabras son buenos candidatos
- Pero algunas palabras no son buenos términos como por ejemplo los artículos pues no tienen contenido
- Se suele tener una lista de palabras que no son buenos términos de indexación
  - STOPLIST
  - Lista de palabras vacias
  - Diccionario negativo



STOPLIST

- Implementación:
  - 1ª aproximación: Palabras más frecuentes
  - Hay que tener cuidado, algunas palabras muy frecuentes tienen significado
    - Ej: Mundo, guerra, sistema
  - En colecciones específicas hay palabras que nos valen:
    - Ej: en una BD informática, ordenador no es un término

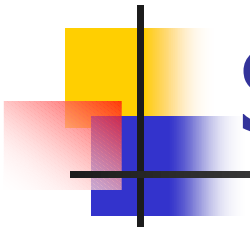


STOPLIST

- Implementación:
  - Tiene pocas palabras
  - Muchos solo tienen 7 (The, and, ...)
  - Rara vez tienen más de 200 palabras

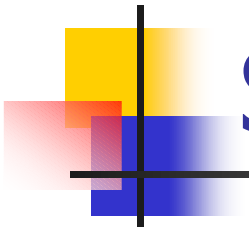
STOPLIST

- Beneficios
  - La indexación es más rápida
  - Las palabras vacías aparecen mucho y su lista de referencias es muy grande
    - Si las quitamos el archivo invertido será más pequeño
    - El archivo invertido se reduce en un 30% ó 40%
  - Aumenta la eficiencia



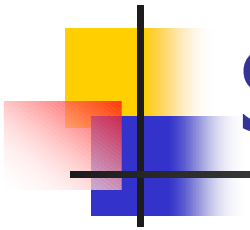
STOPLIST

- Implementación
  - La salida del analizador léxico es comprobada con la Stoplist y se eliminan los términos que aparecen en ella
    - Utilizando un árbol B+
    - Utilizando una tabla Hash
      - Muy rápido
      - La Stoplist es estática



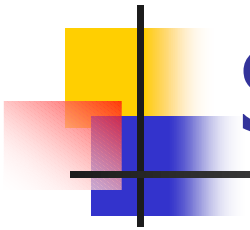
STOPLIST

- Implementación
  - Incorporar la eliminación de las palabras vacías en el analizador léxico
    - Es más eficiente
    - No suele ser necesario en la mayoría de los casos



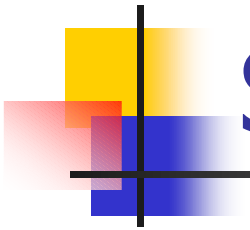
STEMMING

- Consiste en convertir todas las palabras parecidas a una forma común (literalmente ”obtención del tronco”)
- No es hallar la raíz léxica
- Se pretende agrupar términos en un solo término de indexació
  - Eficiencia
  - Eficacia
- Obtención mediante patrones



STEMMING

- Técnicas
  - Búsqueda en una tabla que tiene todas las derivaciones de un término común
    - Sencillo
    - Problemas:
      - Hay que construir la tabla
      - Es difícil para palabras específicas a un dominio

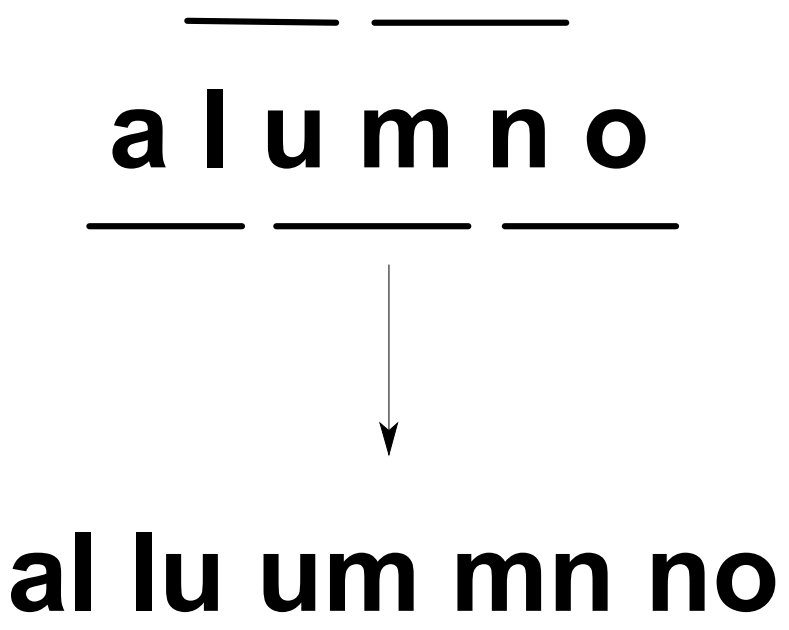


STEMMING

- Técnicas
  - Obtención de la variedad de sucesores
    - Propiedad estructural de la mayoría de los lenguajes
    - Las terminaciones de las palabras siguen determinadas pautas
    - No es necesario construir una tabla pues se construye a partir de una colección
    - Consiste en agrupar palabras con la misma raíz
      - Ej: **disco**, **disco**s, **disco**teca, **disco**grafía

STEMMING

- Técnicas
  - N-gramas
    - No pretende obtener una forma común, si no determinar clases o grupos de términos
    - Es heurístico
    - Se buscan los que comparten un nº mayor de n-gramas

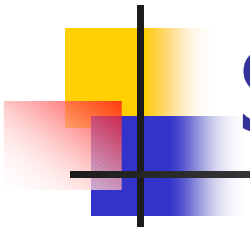


STEMMING

- Técnicas
  - N-gramas
    - Se utiliza un grado de similitud: Coef. DICE
    - Se utiliza un umbral
    - A y B son el nº de n-gramas de las 2 palabras
    - C es el número de n-gramas que comparten

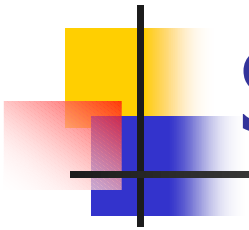
$$S = \frac{2C}{A + B}$$

$$Coeficiente \quad DICE$$



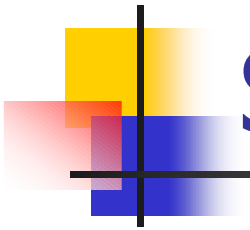
STEMMING

- Técnicas
  - Algoritmos de eliminación de afijos
    - No son reglas heurísticas
    - Son reglas que aplicadas a las palabras nos dan su forma común
    - Se basan en reglas gramaticales aplicadas al revés



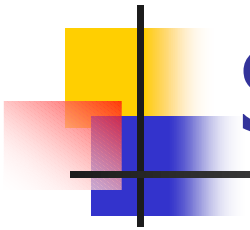
STEMMING

- Técnicas
  - Algoritmos de eliminación de afijos
    - Ventajas
      - Con un número pequeño de reglas obtengo una buena eficiencia
      - Ante una nueva palabra puedo sacar su raíz
    - Desventajas
      - Hay que construir la tabla de reglas
      - Dependen del idioma



STEMMING

- Técnicas
  - Algoritmos de eliminación de afijos
    - El más conocido y usado es el algoritmo de Porter
      - 30 – 40 reglas
      - Sólo elimina sufijos



STEMMING

- Técnicas
  - Algoritmos de eliminación de afijos
    - Sólo suelen eliminarse los sufijos
      - Es más sencillo
      - Hay más sufijos que prefijos
    - No se suelen usar mucho estas técnicas
    - En Web no se usa
      - Hay muchos idiomas
      - La mejora no es claramente importante (quizás porque los estudios sólo se basan en el inglés)