Motores de Busqueda

DAI

Un buscador es un programa de *Recuperación de Información*, que trata de buscar las páginas más relevantes a partir de términos de búsqueda

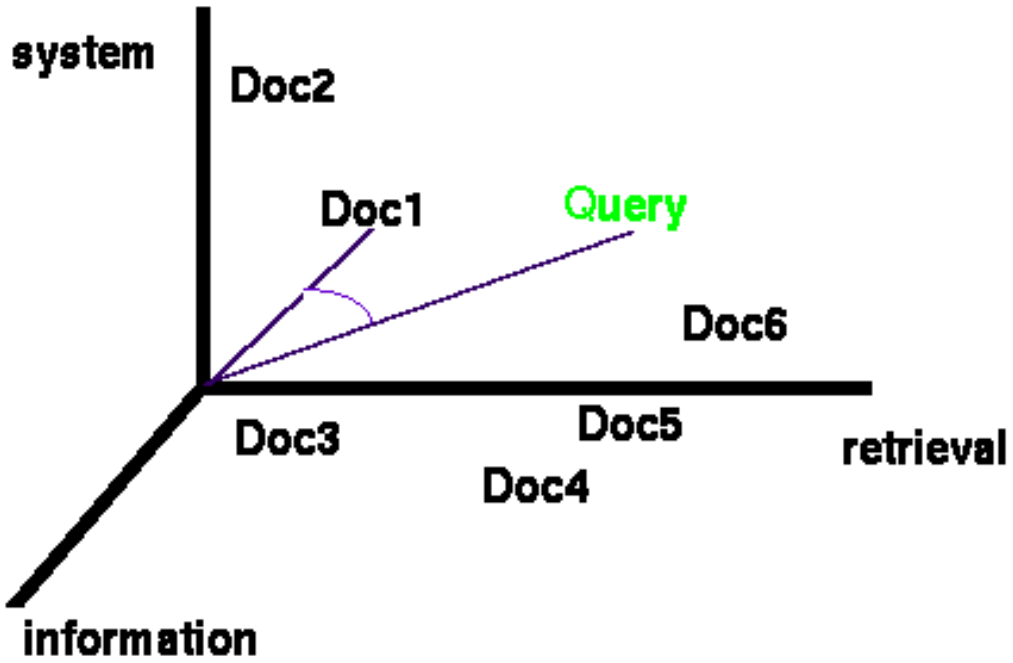
Al contrario que en una base de datos relacional, se busca en **información no estructurada** (texto)

Buscadores

- Una aplicación muy usual en cualquier sitio web
- Rentable `http://www.gurusblog.com/archives/google-presenta-resultados-y-sube-casi-un-5-en-el-after-hours/23/01/2013/`
- Importante, **SEO**
  `http://es.wikipedia.org/wiki/Posicionamiento_en_buscadores`
- ¿Como sería internet sin buscadores?

Modelo de espacio vectorial
Vector Space Model

Se puede modelizar usando un espacio vectorial n-dimensional, en el que cada página sería un punto en este espacio, el texto de la consulta, otro punto. Cuanto más cerca estén un punto de otro más se parecerán



El espacio vectorial

Se asigna a cada página un vector en la que cada coordenada representa la importancia de un posible término de búsqueda (palabra) en la página

El procedimiento sería:

1. Se saca un vocabulario con todas las palabras que aparezcan en cualquier página
2. Se ordenan por orden alfabético
3. A cada palabra se le asigna una dimensión en el espacio vectorial
4. A cada página se le asigna un vector, en el que cada coordenada refleje la importancia de la palabra en la página

mini-ejemplo

```
pagina_1 = '<b>Hola</b>, que haces'

pagina_2 = 'Tienda on-line'


Vocabulario=['haces', 'hola', 'on-line', 'que',
             'tienda']


pag_1 = [1,2,0,1,0]  # 'Hola' está resaltada

pag_2 = [0,0,1,0,1]
```

índice inverso

	haces	*hola*	*on – line*	*que*	*tienda*
pagina_1	1	2	0	1	0
pagina_2	0	0	1	0	1

La tabla se puede leer en horizontal o en vertical

Stop words (palabras vacias)

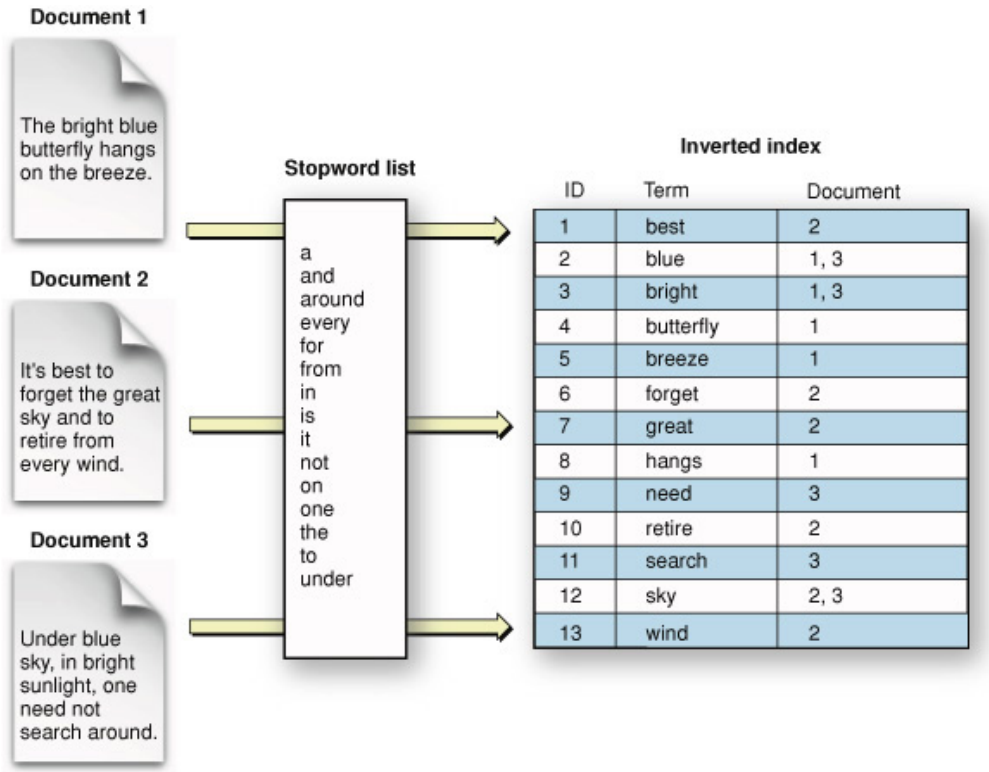
Podríamos quitar palabras sin significado, que no se van a usar en las búsquedas

	haces	*hola*	*on – line*	*tienda*
pagina_1	1	2	0	0
pagina_2	0	0	1	1

Así los índices son más pequeños

índice inverso

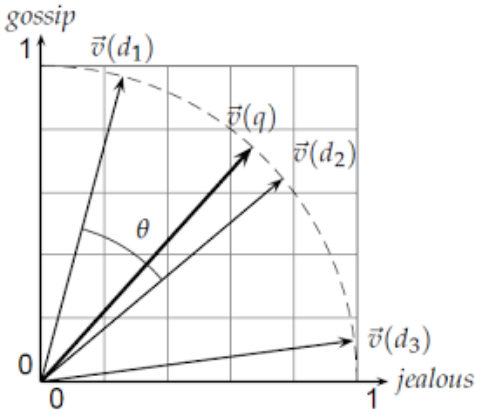
Lo que hacemos es *'invertir'* el texto; en lugar de archivos de páginas con palabras, tenemos índices de palabras con páginas



Métrica

Para computar el parecido de un vector a otro usamos el **coseno** del ángulo que forman

$$similitud(\overrightarrow{a}, \overrightarrow{b}) = cos(\theta) = \frac{\sum a_i b_i}{\|a\| \, \|b\|}$$



Cosine similarity illustrated. $\text{sim}(d_1, d_2) = \cos\theta$.

De esta manera ordenamos los resultados de la búsqueda por relevancia

Producto escalar

Consulta: 'Hola tienda' = $(0, 1, 0, 1)$

Consulta . pagina_1 = $(0, 1, 0, 1) \cdot (1, 2, 0, 0) = 2$

Consulta . pagina_2 = $(0, 1, 0, 1) \cdot (0, 0, 1, 1) = 1$

Peso de las palabras en al página

Para el peso de la palabra en la página podemos tener en cuenta:

- El número de veces que se repite la palabra en la página
- El lugar que ocupe (las palabras en los títulos o encabezados serán más importantes)
- El resaltado (palabras en negrita, etc)

En todo caso esté número tiene que estar **normalizado**, es decir cada página tiene el mismo peso que reparte entre las palabras que aparecen en ella.

Peso de las palabras: idf
Inverse Document Frecuency

No todas las palabras en una consulta deben tener la misma importancia en la búsqueda.

pe. en la búsqueda `'Restaurantes de Granada'`

buscamos páginas en las que sea importante la palabra '**Restaurantes**' y la palabra '**Granada**',
pero no la palabra *'de'*

Peso de las palabras: idf

Inverse Document frecuency

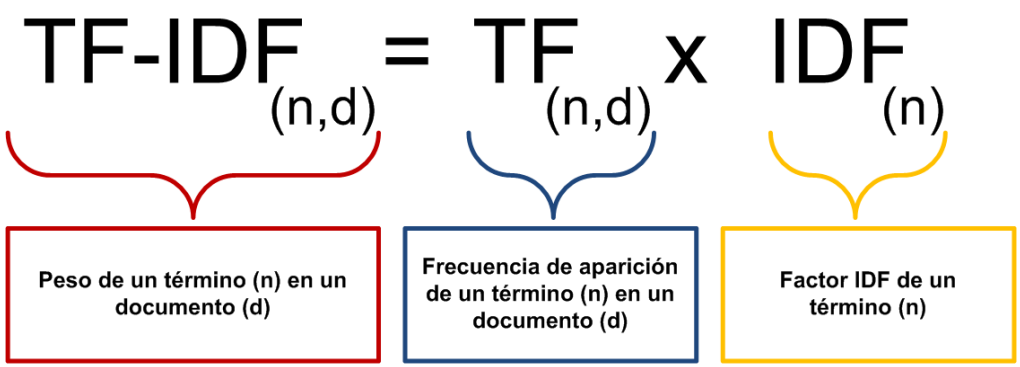
Esto se corrige aplicando un peso a las palabras de la búsqueda **Frecuencia inversa por documento**

$$idf = \log \frac{N}{n_i}$$

N es el número total de páginas
n_i es el número de páginas donde aparece la palabra i

Las palabras comunes, *(stop words)*, tendrán un `idf`, cercano a 0, y por tanto podemos quitarlas del índice para ahorrar espacio y tiempo de proceso

Al final tenemos el esquema de pesos tf-idf



http://en.wikipedia.org/wiki/Tf%E2%80%93idf

Seudocódigo

```
relevancia = {}

for t in palabras_busqueda:
    idf = IDF_DE(t)
    lista_pag = PAGS_DE(t)

    for pag lista_pag:
        relevancia[pag] += PUNT_EN(t, pag) * idf;
    }
}
```

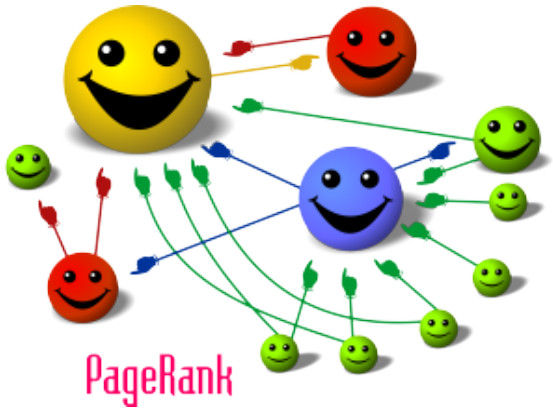
Otro truco para ahorrar espacio y tiempo en las búsquedas indexar el morfema raíz de las palabras, quitando el morfema final

P.e. podemos indexar `niñ`, cuando aparezca *niño, niños, niña, o niñas*

Así el índice es más pequeño y al buscar *niño*, también recuperamos páginas en las que aparece *niña o niños*

Peso de las páginas
Page Rank

También podemos tener en cuenta para el que unas páginas sean más relevantes que otras



Google utiliza un algoritmo en el que se tienen en cuenta los enlaces hacía cada página; una pagina será más importante si otras páginas importantes la enlazan

`http://es.wikipedia.org/wiki/PageRank`
`http://www.mipagerank.com/`