Search – indexing the web

Find documents in response to the user’s query

Information retrieval

• find documents
• find documents in response to user query
• find relevant documents in response to user query
• quickly find relevant documents in response to user query
Key steps

- Collect and index documents
- Interpret user query
- Find documents that may be relevant
- Present most relevant documents first

Indexing Process
Key Steps

• Collect and **index** documents
• Interpret user query
• Find documents that may be relevant
• Present most relevant documents first

What makes search engines fast?

*The index!*

• in a book : words $\rightarrow$ pages
  hundreds of words
hundreds of pages
• in a library : topics/author/title $\rightarrow$ books
tens of thousands of topics
millions of books
• on the web : words $\rightarrow$ documents
  hundreds of thousands of words
billions of documents
indexing the web
with thanks to Victor Lavrenko (& Dr. Seuss)

documents
D1: He likes to wink, he likes to drink.
D2: He likes to drink and drink and drink.
D3: The thing he likes to drink is ink.
D4: The ink he likes to drink is pink.
D5: He likes to wink and drink pink ink.

vocabulary
he
drink
ink
likes
pink
thing
wink

remove stop words

Some words are so common they aren’t useful for indexing. In this example, we remove the ‘stop words’
-> ‘to’ ‘and’ ‘the’ ‘is’
Then we just count the words in each document
We ignore the linguistic structure and just count words. This is very simplistic – but it works!

355 another beating Dow falls points takes
  Dow takes another beating, falls 355 points.

fat fries French MacDonalds obesity said
  does ‘French’ refer to France here?
indexing the web

- one entry per word
- number times word in document

<table>
<thead>
<tr>
<th></th>
<th>he</th>
<th>drink</th>
<th>ink</th>
<th>likes</th>
<th>pink</th>
<th>thing</th>
<th>wink</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D2</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

- "Inverted Index": for each word, gives set of documents where it occurred

<table>
<thead>
<tr>
<th></th>
<th>he</th>
<th>drink</th>
<th>ink</th>
<th>likes</th>
<th>pink</th>
<th>thing</th>
<th>wink</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D2</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

- He likes to wink, he likes to drink.
- He likes to drink and drink and drink.
- The thing he likes to drink is ink.
- The ink he likes to drink is pink.
- He likes to wink and drink pink ink.
indexing the web

<table>
<thead>
<tr>
<th>he</th>
<th>drink</th>
<th>ink</th>
<th>likes</th>
<th>pink</th>
<th>wink</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

D1: He likes to wink, he likes to drink.
D2: He likes to drink and drink and drink.
D3: The thing he likes to drink is ink.
D4: The ink he likes to drink is pink.
D5: He likes to wink and drink pink ink.

But we’re wasting A LOT of space!

Inverted lists are very sparse. Look at the entry for “thing”. It’s only in ONE document!

Remember, documents are just bags of words

Use a sparse representation:
For each word, make a list of tuples containing (document ID, Frequency of word)
Sorted by words

Advantages:
compact
easy to use to find documents that contain specific words
indexing the web

The sparse representation is much more compact

look at the entry for “thing”

using the index

such information can be used to calculate relevance
building the index

MAP: different documents are processed by different computers to produce bags of words

MAP: it is easy to produce the index for one document at a time
building the index

MAP: different computers can do this for different documents

<table>
<thead>
<tr>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>D4</th>
<th>D5</th>
</tr>
</thead>
<tbody>
<tr>
<td>he</td>
<td>he</td>
<td>he</td>
<td>he</td>
<td>he</td>
</tr>
<tr>
<td>drink</td>
<td>drink</td>
<td>drink</td>
<td>drink</td>
<td>drink</td>
</tr>
<tr>
<td>ink</td>
<td>ink</td>
<td>ink</td>
<td>ink</td>
<td>ink</td>
</tr>
<tr>
<td>likes</td>
<td>likes</td>
<td>likes</td>
<td>likes</td>
<td>likes</td>
</tr>
<tr>
<td>pink</td>
<td>pink</td>
<td>pink</td>
<td>pink</td>
<td>pink</td>
</tr>
<tr>
<td>thing</td>
<td>thing</td>
<td>thing</td>
<td>thing</td>
<td>thing</td>
</tr>
<tr>
<td>wink</td>
<td>wink</td>
<td>wink</td>
<td>wink</td>
<td>wink</td>
</tr>
</tbody>
</table>

building the index

MAP: different computers can do this for different collections of documents

<table>
<thead>
<tr>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>D4</th>
<th>D5</th>
</tr>
</thead>
<tbody>
<tr>
<td>he</td>
<td>he</td>
<td>he</td>
<td>he</td>
<td>he</td>
</tr>
<tr>
<td>drink</td>
<td>drink</td>
<td>drink</td>
<td>drink</td>
<td>drink</td>
</tr>
<tr>
<td>ink</td>
<td>ink</td>
<td>ink</td>
<td>ink</td>
<td>ink</td>
</tr>
<tr>
<td>likes</td>
<td>likes</td>
<td>likes</td>
<td>likes</td>
<td>likes</td>
</tr>
<tr>
<td>pink</td>
<td>pink</td>
<td>pink</td>
<td>pink</td>
<td>pink</td>
</tr>
<tr>
<td>thing</td>
<td>thing</td>
<td>thing</td>
<td>thing</td>
<td>thing</td>
</tr>
<tr>
<td>wink</td>
<td>wink</td>
<td>wink</td>
<td>wink</td>
<td>wink</td>
</tr>
</tbody>
</table>
building the index

REDUCE: different computers share the work of merging the index one word at a time

D1 + D3
he [1:2] [3:1]
drink [1:1] [3:1]
likes [1:2] [3:1]
pink [3:1]
thing [3:1]
wink [1:1]

D2 + D4
he [2:1] [4:1]
drink [2:3] [4:1]
ink [4:1]
likes [2:1] [4:1]
pink [4:1]
thing [4:1]
wink

D5

he [1:2] [2:1] [3:1] [4:1]
drink [1:1] [2:3] [3:1] [4:1]
ink [3:1] [4:1]
likes [1:2] [2:1] [3:1] [4:1]
pink [4:1]
thing [3:1]
wink [1:1]
Building the Index

**REDUCE**: different computers can share the work of merging the index one word at a time

```
(D1 + D3) + (D2 + D4) + D5
```

**Making it efficient**

**Divide and Conquer**

**MAP**: it is easy to produce the index for one document at a time

**REDUCE**: different computers can share the work of merging the index – and different computers can work on different words
using the index

<table>
<thead>
<tr>
<th>ink</th>
<th>[3:1][4:1][5:1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>wink</td>
<td>[1:2][5:1]</td>
</tr>
<tr>
<td>ink AND wink</td>
<td>[5:1(1,1)]</td>
</tr>
<tr>
<td>ink OR wink</td>
<td>[1:0.2][3:1(0,0)][4:1(0,0)][5:1(1,1)]</td>
</tr>
</tbody>
</table>

- different computers can provide information for different query words
- this information can be combined to calculate relevance