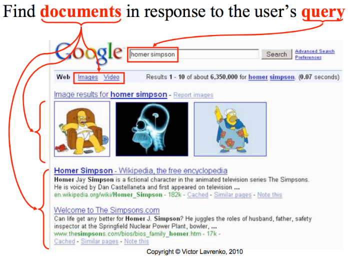
Search – indexing the web

Find **documents** in response to the user's **query**



Results **1** - **10** of about **6,350,000** for **homer simpson**. (**0.07** seconds)

Image results for **homer simpson** - Report images

Homer Simpson - Wikipedia, the free encyclopedia
Homer Jay **Simpson** is a fictional character in the animated television series The Simpsons. He is voiced by Dan Castellaneta and first appeared on television **...**
en.wikipedia.org/wiki/**Homer_Simpson** - 182k - Cached - Similar pages - Note this

Welcome to The Simpsons.com
Can life get any better for **Homer** J. **Simpson**? He juggles the roles of husband, father, safety inspector at the Springfield Nuclear Power Plant, bowler, **...**
www.the**simpsons**.com/bios/bios_family_**homer**.htm - 17k -
Cached - Similar pages - Note this

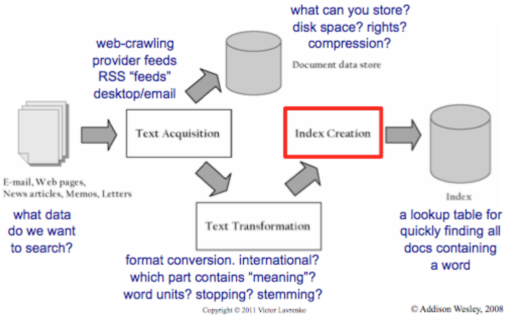
information retrieval

- 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



what can you store?
disk space? rights?
compression?

web-crawling
provider feeds
RSS “feeds”
desktop/email

Document data store

Text Acquisition

Index Creation

E-mail, Web pages,
News articles, Memos, Letters

what data
do we want
to search?

Text Transformation

Index

a lookup table for
quickly finding all
docs containing
a word

format conversion. international?
which part contains “meaning”?
word units? stopping? stemming?

Copyright © 2011 Victor Lavrenko

© Addison Wesley, 2008

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 → pages
  hundreds of words
  hundreds of pages
- in a library : topics/author/title → books
  tens of thousands of topics
  millions of books
- on the web : words → 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

a
about
above
after
again
against
all
am
an
and
any
are
aren't
as
at
be
because
been
before
being
below
between
both
but
by
can't
cannot
could
couldn't

did
didn't
do
does
doesn't
doing
don't
down
during
each
few
for
from
further
had
hadn't
has
hasn't
have
haven't
having
he
he'd
he'll
he's
her
here
here's
hers

herself
him
himself
his
how
how's
i
i'd
i'll
i'm
i've
if
in
into
is
isn't
it
it's
its
itself
let's
me
more
most
mustn't
my
myself
no
nor

not
of
off
on
once
only
or
other
ought
our
ours
ourselves
out
over
own
same
shan't
she
she'd
she'll
she's
should
shouldn't
so
some
such
than
that
that's

the
their
theirs
them
themselves
then
there
there's
these
they
they'd
they'll
they're
they've
this
those
through
to
too
under
until
up
very
was
wasn't
we
we'd
we'll
we're

we've
were
weren't
what
what's
when
when's
where
where's
which
while
who
who's
whom
why
why's
with
won't
would
wouldn't
you
you'd
you'll
you're
you've
your
yours
yourself
yourselves

Bag-of-words

- 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

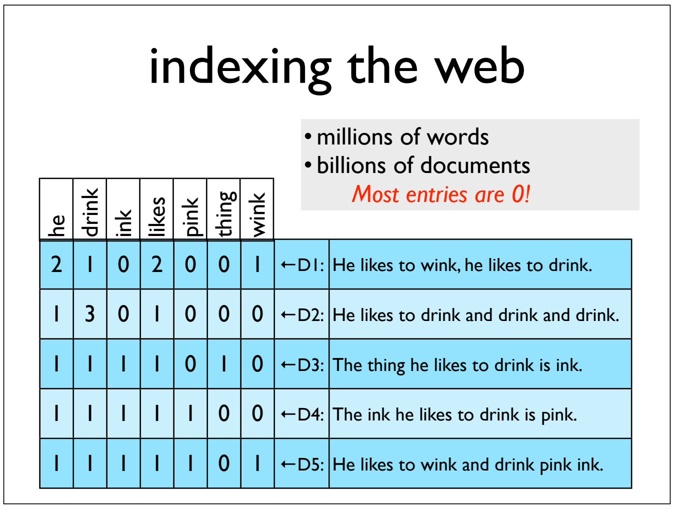
- one entry per word
- number times word in document

he	drink	ink	likes	pink	thing	wink		
2	1	0	2	0	0	1	←D1:	He likes to wink, he likes to drink.
1	3	0	1	0	0	0	←D2:	He likes to drink and drink and drink.
1	1	1	1	0	1	0	←D3:	The thing he likes to drink is ink.
1	1	1	1	1	0	0	←D4:	The ink he likes to drink is pink.
1	1	1	1	1	0	1	←D5:	He likes to wink and drink pink ink.

indexing the web

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

he	drink	ink	likes	pink	thing	wink		
2	1	0	2	0	0	1	←D1:	He likes to wink, he likes to drink.
1	3	0	1	0	0	0	←D2:	He likes to drink and drink and drink.
1	1	1	1	0	1	0	←D3:	The thing he likes to drink is ink.
1	1	1	1	1	0	0	←D4:	The ink he likes to drink is pink.
1	1	1	1	1	0	1	←D5:	He likes to wink and drink pink ink.



indexing the web

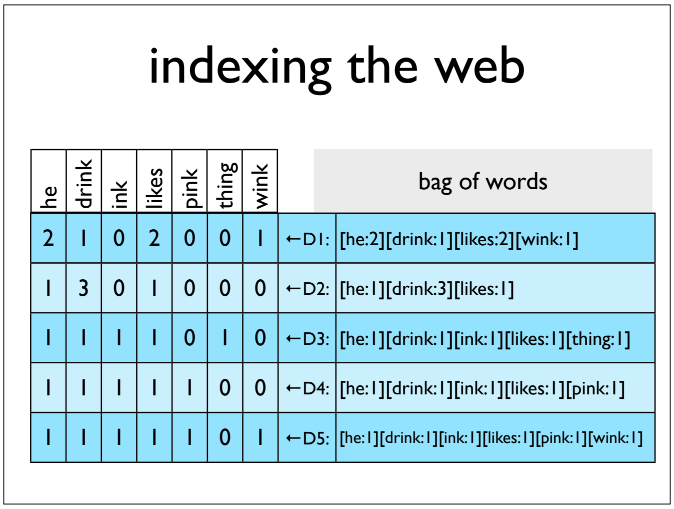
- millions of words
- billions of documents

Most entries are 0!

he	drink	ink	likes	pink	thing	wink		
2	1	0	2	0	0	1	←D1:	He likes to wink, he likes to drink.
1	3	0	1	0	0	0	←D2:	He likes to drink and drink and drink.
1	1	1	1	0	1	0	←D3:	The thing he likes to drink is ink.
1	1	1	1	1	0	0	←D4:	The ink he likes to drink is pink.
1	1	1	1	1	0	1	←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!



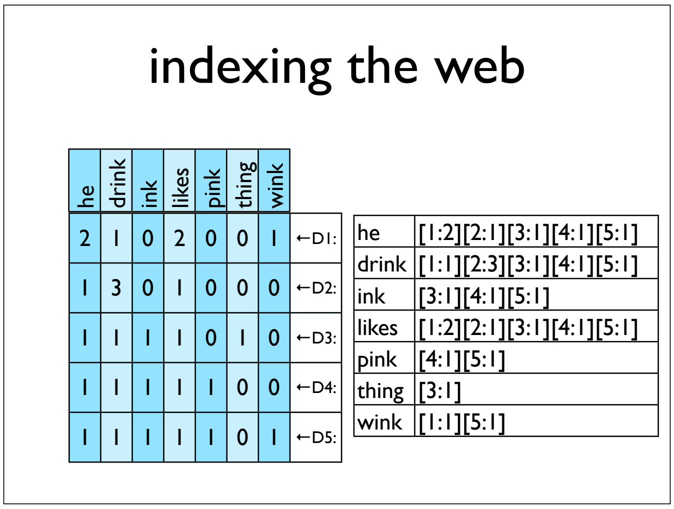
indexing the web

he	drink	ink	likes	pink	thing	wink		bag of words
2	1	0	2	0	0	1	←D1:	[he:2][drink:1][likes:2][wink:1]
1	3	0	1	0	0	0	←D2:	[he:1][drink:3][likes:1]
1	1	1	1	0	1	0	←D3:	[he:1][drink:1][ink:1][likes:1][thing:1]
1	1	1	1	1	0	0	←D4:	[he:1][drink:1][ink:1][likes:1][pink:1]
1	1	1	1	1	0	1	←D5:	[he:1][drink:1][ink:1][likes:1][pink:1][wink:1]

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



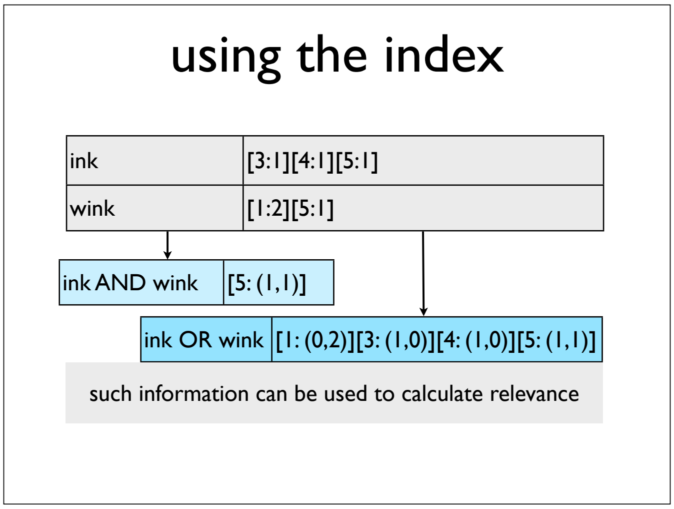
indexing the web

he	drink	ink	likes	pink	thing	wink	
2	1	0	2	0	0	1	←D1:
1	3	0	1	0	0	0	←D2:
1	1	1	1	0	1	0	←D3:
1	1	1	1	1	0	0	←D4:
1	1	1	1	1	0	1	←D5:

he	[1:2][2:1][3:1][4:1][5:1]
drink	[1:1][2:3][3:1][4:1][5:1]
ink	[3:1][4:1][5:1]
likes	[1:2][2:1][3:1][4:1][5:1]
pink	[4:1][5:1]
thing	[3:1]
wink	[1:1][5:1]

The sparse representation is much more compact

look at the entry for “thing”



using the index

ink	[3:1][4:1][5:1]
wink	[1:2][5:1]

ink AND wink	[5: (1,1)]

ink OR wink	[1: (0,2)][3: (1,0)][4: (1,0)][5: (1,1)]

such information can be used to calculate relevance

building the index

he	drink	ink	likes	pink	thing	wink	MAP : different documents are processed by different computers to produce bags of words	
2	1	0	2	0	0	1	←D1:	[he:2][drink:1][likes:2][wink:1]
1	3	0	1	0	0	0	←D2:	[he:1][drink:3][likes:1]
1	1	1	1	0	1	0	←D3:	[he:1][drink:1][ink:1][likes:1][thing:1]
1	1	1	1	1	0	0	←D4:	[he:1][drink:1][ink:1][likes:1][pink:1]
1	1	1	1	1	0	1	←D5:	[he:1][drink:1][ink:1][likes:1][pink:1][wink:1]

building the index

2	1	0	2	0	0	1	←D1:	[he:2][drink:1][likes:2][wink:1]

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

he	[1:2]
drink	[1:1]
ink	
likes	[1:2]
pink	
thing	
wink	[1:1]

building the index

MAP : different computers can do this for different documents

D1		D2		D3		D4		D5	
he	[1:2]	he	[2:1]	he	[3:1]	he	[4:1]	he	[5:1]
drink	[1:1]	drink	[2:3]	drink	[3:1]	drink	[4:1]	drink	[5:1]
ink		ink		ink	[3:1]	ink	[4:1]	ink	[5:1]
likes	[1:2]	likes	[2:1]	likes	[3:1]	likes	[4:1]	likes	[5:1]
pink		pink		pink		pink	[4:1]	pink	[5:1]
thing		thing		thing	[3:1]	thing		thing	
wink	[1:1]	wink		wink		wink		wink	[5:1]

building the index

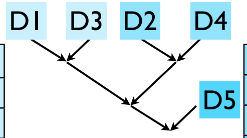
MAP : different computers can do this for different collections of documents

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

D2		D4		D5	
he	[2:1]	he	[4:1]	he	[5:1]
drink	[2:3]	drink	[4:1]	drink	[5:1]
ink		ink	[4:1]	ink	[5:1]
likes	[2:1]	likes	[4:1]	likes	[5:1]
pink		pink	[4:1]	pink	[5:1]
thing		thing		thing	
wink		wink		wink	[5:1]

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]
ink	[3:1]
likes	[1:2] [3:1]
pink	
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	
wink	

building the index

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

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

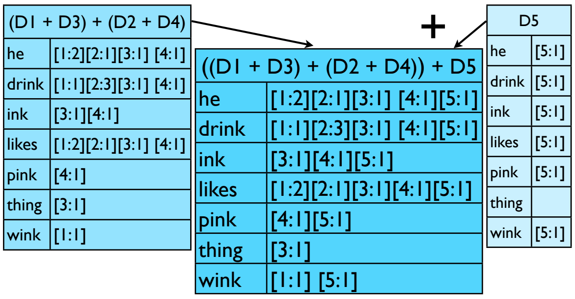
+

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

(D1 + D3) + (D2 + D4)	
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)	
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]

((D1 + D3) + (D2 + D4)) + D5	
he	[1:2][2:1][3:1] [4:1][5:1]
drink	[1:1][2:3][3:1] [4:1][5:1]
ink	[3:1][4:1][5:1]
likes	[1:2][2:1][3:1][4:1][5:1]
pink	[4:1][5:1]
thing	[3:1]
wink	[1:1] [5:1]

D5	
he	[5:1]
drink	[5:1]
ink	[5:1]
likes	[5:1]
pink	[5:1]
thing	
wink	[5:1]

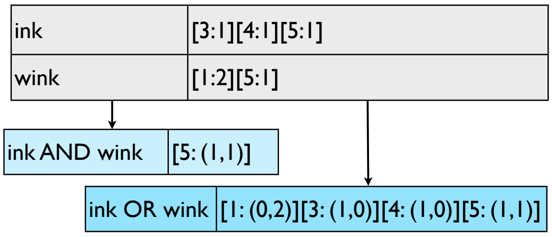
Making it efficient

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



ink	[3:1][4:1][5:1]
wink	[1:2][5:1]

ink AND wink	[5: (1,1)]
ink OR wink	[1: (0,2)][3: (1,0)][4: (1,0)][5: (1,1)]

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