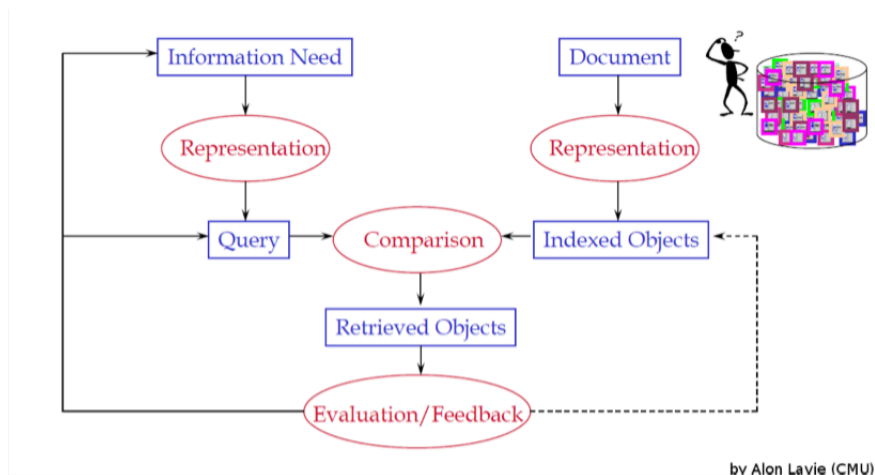


Information Retrieval

Task definition

Google's IR system: Finding pages that contain the words in the query. Rank by relevance to the query. By clever indexing(and hardware) so fast.

Text Retrieval: Find document that are relevant to a user query.



Given: A large, static **document** collection Given: An information need(keyword-based **query**)

Task: Find all and only documents **relevant** to query

Typical IR systems

- Search a set of abstracts
- Search newspaper articles
- Library search
- Search the Web

Typically, more statistics than language, but the object to retrieve(and process) is language. 通常情况下统计比语言更多，但是要检索和处理的对象是语言

Formulate a query: **Query type:** normally keywords, could be natural language.

Document are represented as **indexing**.

By **retrieval model** can the system find then best-matching document.

How does the system find it **efficiently**?

Result are represented to users as **unsorted list/ ranked list/ clusters**

Evaluation to tell whether the system is good.

Document Indexing

Manual Approaches

Indexing by **humans**(using fixed **vocabularies**)

Labour and training intensive

Large vocabularies(several thousand items)

- Dewey Decimal System
- Library of Congress Subject Headings
- ACM - subfields of CS (ACM Computing Systems(1998))
- Mesh - Medical Subject Headings
 - a very large controlled vocabulary for describing/indexing medical documents, e.g. journal papers and books
 - provides a hierarchy of descriptors (a.k.a. subject headings)
 - assigned to documents to describe their content
 - hierarchy has a number of top-level categories, e.g.:
 - Anatomy [A]
 - Organisms [B]
 - Diseases [C]
 - Chemicals and Drugs [D]
 - Analytical, Diagnostic and Therapeutic Techniques and Equipment [E]
 - Psychiatry and Psychology [F]
 - Biological Sciences [G]
 - And a number of subcategories (more specific/detailed terms):
 - And a number of subsubcategories (even more specific/detailed terms):
 - And a number of subsubsubcategories (yet again more specific/ detailed terms): 哦吼吼吼
- MEDLINE — Medical Literature Analysis and Retrieval System Online
 - international database of literature for medicine and the life sciences
 - includes papers from ≈5600 different sources (mostly journals), in various languages
 - database now holds records for ≈26 million papers
- Each MEDLINE article indexed with 10-15 descriptors from MeSH
 - papers accessed by PubMed search engine interface, using MeSH terms (and other terms, e.g. author name, etc)
 - by default, all descriptors below a given one in the hierarchy are also included in search

Advantages of manual indexing:

- **High precision** searches
- Word well for **closed collections**(books in a library)

Problems:

- Searchers need to **know terms** to achieve high precision
- Labellers need to be **trained** to achieve consistency
 - Not feasible to expect this from all content creators on the web
- Collections are **dynamic** -> schemes change constantly

Automatic Approaches

Term manipulation(certain words count as the same term)

Term weighting(certain terms are more important than others)

Index terms must only derive from text

Automatic indexing:

- No predefined set of index terms
- Instead: use **natural language** as indexing language
- Words in the document give information about its content
- Implementation of indices: **inverted files**
- This is what Google's IR system does
 - at least, it's an important **part** of the story

EXAMPLE:

A small collection of documents:

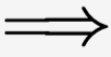
<i>Document</i>	<i>Text</i>
1	Pease porridge hot , pease porridge cold
2	Pease porridge in the pot
3	Nine days old
4	Some like it hot , some like it cold
5	Some like it in the pot
6	Nine days old

Say we want to search for word **hot**. How do we do it?

A basic inverted file index:

- Records for each term, the **ids** of the documents in which it appears
- Only matters id it does or does not **appear** - not how many times

<i>Doc</i>	<i>Text</i>
1	Pease porridge hot, pease porridge cold
2	Pease porridge in the pot
3	Nine days old
4	Some like it hot, some like it cold
5	Some like it in the pot
6	Nine days old

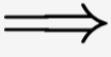


<i>Num</i>	<i>Token</i>	<i>Docs</i>
1	cold	1, 4
2	days	3, 6
3	hot	1, 4
4	in	2, 5
5	it	4, 5
6	like	4, 5
7	nine	3, 6
8	old	3, 6
9	pease	1, 2
10	porridge	1, 2
11	pot	2, 5
12	some	4, 5
13	the	2, 5

A more sophisticated version:

- also record **count of occurrences** within each document
- Help find documents more relevant to query

<i>Doc</i>	<i>Text</i>
1	Pease porridge hot, pease porridge cold
2	Pease porridge in the pot
3	Nine days old
4	Some like it hot, some like it cold
5	Some like it in the pot
6	Nine days old

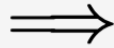


<i>Num</i>	<i>Token</i>	<i>Docs</i>
1	cold	1:1, 4:1
2	days	3:1, 6:1
3	hot	1:1, 4:1
4	in	2:1, 5:1
5	it	4:2, 5:1
6	like	4:2, 5:1
7	nine	3:1, 6:1
8	old	3:1, 6:1
9	pease	1:2, 2:1
10	porridge	1:2, 2:1
11	pot	2:1, 5:1
12	some	4:2, 5:1
13	the	2:1, 5:1

A more sophisticated version:

- Also record **position** of each term occurrence within documents
- maybe useful for searching for **phrases** in documents

<i>Doc</i>	<i>Text</i>
1	Pease porridge hot, pease porridge cold
2	Pease porridge in the pot
3	Nine days old
4	Some like it hot, some like it cold
5	Some like it in the pot
6	Nine days old



<i>Num</i>	<i>Token</i>	<i>Docs</i>
1	cold	1:(6), 4:(8)
2	days	3:(2), 6:(2)
3	hot	1:(3), 4:(4)
4	in	2:(3), 5:(4)
5	it	4:(3, 7), 5:(3)
6	like	4:(2, 6), 5:(2)
7	nine	3:(1), 6:(1)
8	old	3:(3), 6:(3)
9	pease	1:(1, 4), 2:(1)
10	porridge	1:(2, 5), 2:(2)
11	pot	2:(5), 5:(6)
12	some	4:(1, 5), 5:(1)
13	the	2:(4), 5:(5)

Automated Retrieval Models

Bag-of-Words Approach:

- Standard approach to representing documents (and queries) in IR:
 - Record what words(terms) are present
 - Usually, plus count of term in each document
- Ignores relations between words
 - i.e. of order, proximity, etc
 - e.g. rabbit eating = eating rabbit
- such representations known as bag-of words
 - c.f. mathematical structure "bag" -- like a set (i.e. unordered), but records a count for each element

Boolean Model

Boolean search:

- Binary decision: is documents relevant or not?
- Presence of term is necessary and sufficient for match
- Boolean operators are set operations (AND,OR)

Approach:

Construct complex search commands, by

- combineing bvasic search terms (keywords)
- using boolean operators (AND,OR,NOT,BUT,XOR)
- E.g. Monte-Carlo AND (importance OR stratification) BUT gambling

- Boolean query provides a simple logical basis for deciding whether any document should be returned, based on:
 - whether basic terms of query do/do not appear in the document
 - the meaning of the logical operators

Boolean operators have a set-theoretic interpretation for efficient retrieval

Overall document collection forms maximal documents set

Let $d(E)$ denote the document set for expression E

- E either a basic term or boolean expression

Boolean operators map to set-theoretic operations:

- AND $\rightarrow \cap$ (intersection): $d(E_1 \text{ AND } E_2) = d(E_1) \cap d(E_2)$
- OR $\rightarrow \cup$ (union): $d(E_1 \text{ OR } E_2) = d(E_1) \cup d(E_2)$
- NOT $\rightarrow \complement$ (complement): $d(\text{NOT } E) = d(E)^\complement$
- BUT $\rightarrow -$ (difference): $d(E_1 \text{ BUT } E_2) = d(E_1) - d(E_2)$

Summary:

Document either match or don't match

- expert knowledge needed to create high-precision queries \rightarrow ok for expert users
- often used by bibliographic search engines(library)

Not good for the majority of users:

- Most users not familiar with writing Boolean queries \rightarrow not natural
- Most users don't want to wade through 1000s unranked result lists \rightarrow unless very specific search in small collections
- This is particularly true of web search \rightarrow large set of docs

Ranked Retrieval Methods

Ranked algorithms:

- Frequency of document terms
- not all search terms necessarily present in document
- Incarnations:
 - The Vector Space Model (SMART, Salton et al, 1971)
 - The probabilistic model (OKAPI, Robertson/Sparck Jones, 1976)
 - Web Search Engines

The Vector Space Model

Documents are also represented as bag of words

Documents are points in high-dimensional vector space

- each term in index is a dimension \rightarrow sparse vectors

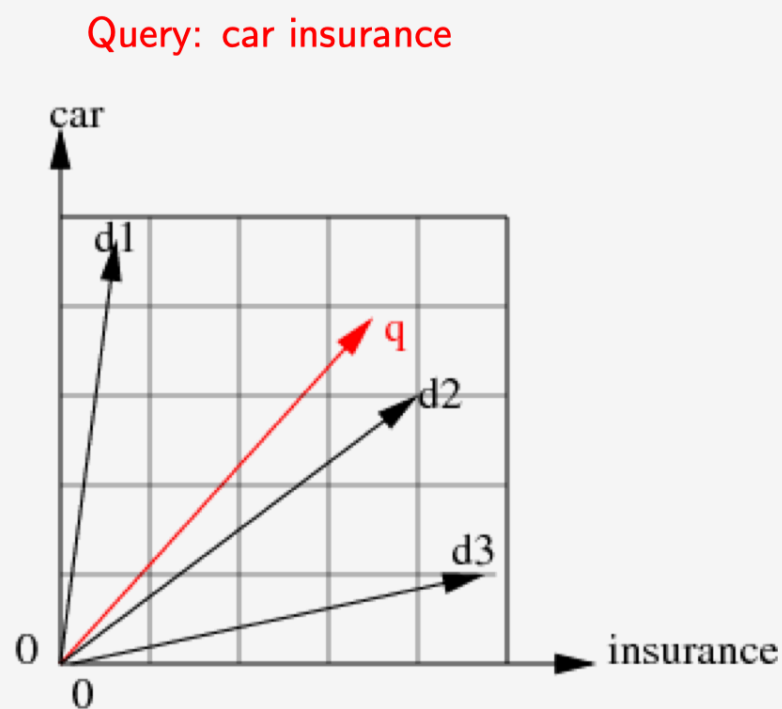
- Values are frequencies of terms in documents, or variants of frequency

Queries are also represented as vectors (for terms that exist in index)

Approach:

- Select documents(s) with highest document-query similarity
- Document-query similarity is a model for relevance(ranking)
- With ranking, the number of returned documents is less relevant --> users start at the top and stop when satisfied

2 dimensions:



Approach: compare vector of query against vector of each document

- to rank documents according to their similarity to the query

	Term ₁	Term ₂	Term ₃	...	Term _n
Doc ₁	9	0	1	...	0
Doc ₂	0	1	0	...	10
Doc ₃	0	1	0	...	2
...
Doc _N	4	7	0	...	5
Q	0	1	0	...	1

How to measure similarity between vectors?

- Each document and the query are represented as a vector of n values:

$$\vec{d}^i = (d_1^i, d_2^i, \dots, d_n^i), \vec{q} = (q_1, q_2, \dots, q_n)$$

- Many metrics of similarity between 2 vectors, e.g.: Euclidean

$$\sqrt{\sum_{k=1}^n (q_k - d_k)^2}$$

E.g.: Distance between:

$$Doc_1 \text{ and } Q = \sqrt{(9-0)^2 + (0-1)^2 + (1-0)^2 + (0-1)^2} = \sqrt{84} = 9.15$$

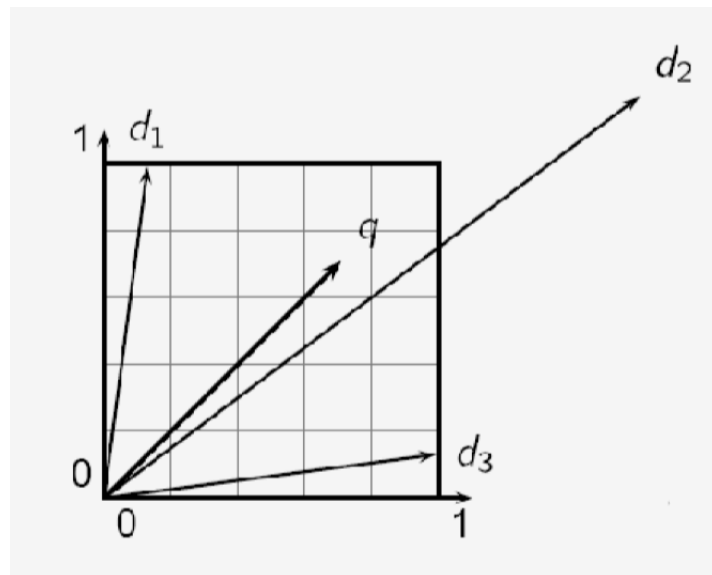
$$Doc_2 \text{ and } Q = \sqrt{(0-0)^2 + (1-1)^2 + (0-0)^2 + (10-1)^2} = \sqrt{81} = 9$$

$$Doc_3 \text{ and } Q = \sqrt{(0-0)^2 + (1-1)^2 + (0-0)^2 + (2-1)^2} = \sqrt{1} = 1$$

Doc 3 is the closest (shortest distance)

Is it a good idea? using Euclidean?

- distance is large for vectors of different lengths, even if by only one term (e.g. Doc₂ and Q)
- means frequency of terms given too much impact



- Better similarity metric, used in vector-space model: **cosine** of the angle between two vectors \vec{x} and \vec{y}

$$\cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{|\vec{x}| |\vec{y}|} = \frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum_{i=1}^n x_i^2} \sqrt{\sum_{i=1}^n y_i^2}}$$

- It can be interpreted as the normalized correlation coefficient:

i.e. it computes how well the x_i and y_i correlate, and then divides by the length of the vectors, to scale for their magnitude

- the vector \vec{x} is normalized by dividing its components by its length:

$$|\vec{x}| = \sqrt{\sum_{i=1}^n x_i^2}$$

- The cosine value ranges from:

- 1, for vectors pointing in the same direction, to
- 0, for orthogonal vectors, to
- 1, for vectors point in opposite directions

- Specialising the equation to comparing a query q and document s :

$$\text{sim}(\vec{q}, \vec{d}) = \cos(\vec{q}, \vec{d}) = \frac{\sum_{i=1}^n q_i d_i}{\sqrt{\sum_{i=1}^n q_i^2} \sqrt{\sum_{i=1}^n d_i^2}}$$

i.e. computes how well occurrences of each term i correlate in query and document, then scales for the magnitude of the overall vectors

Term Manipulation

What counts as a term?

Common to just use the words, but pre-process them for generalisation

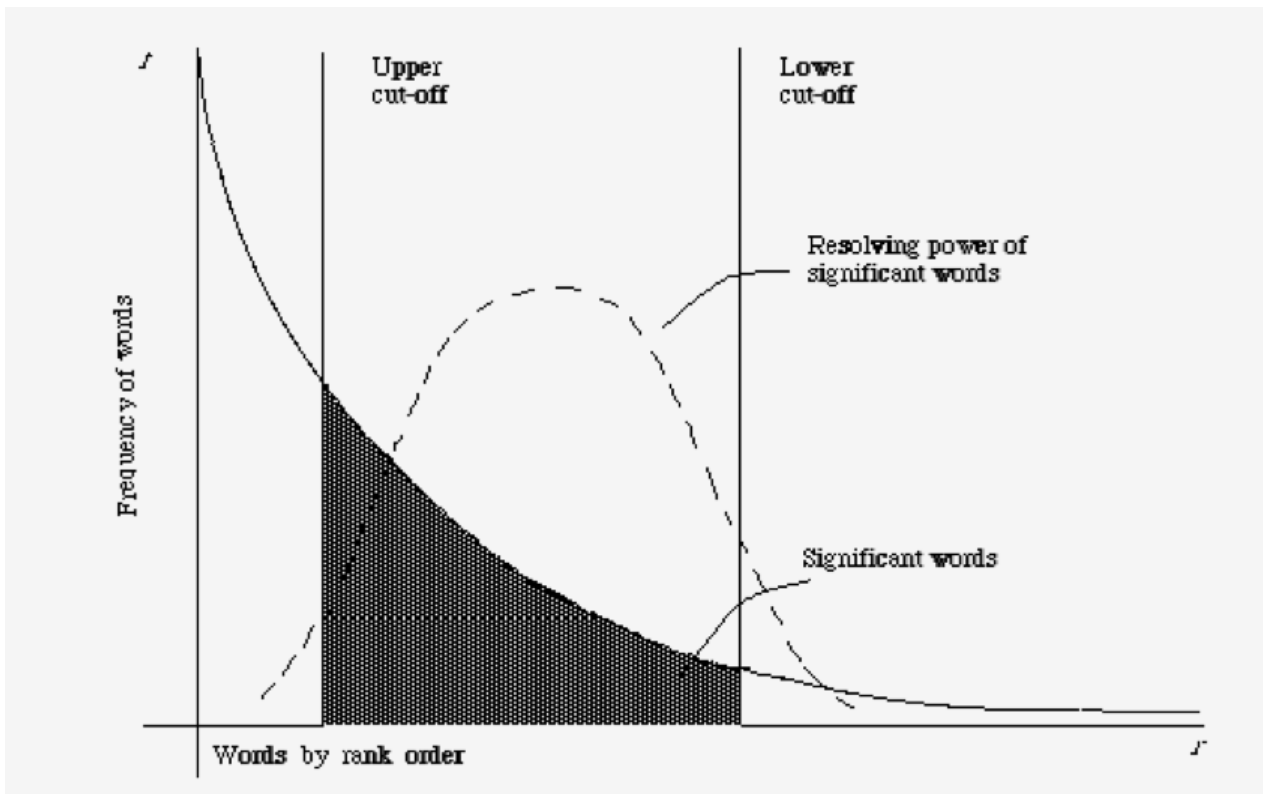
- **Tokenization:** split words from punctuation (get rid of punctuation)
e.g. word-based. → word based
three issues: → three issues
- **Capitalisation:** normalise all words to lower (or upper) case
e.g. Cat and cat should be seen as the same term, but should we conflate Turkey and turkey?
- **Lemmatisation:** conflate different inflected forms of a word to their basic form (singular, present tense, 1st person):
e.g. cats, cat → cat; have, has, had → have; worried, worries → worry
- **Stemming:** conflate morphological variants by chopping their affix:

CONNECT	WORRY	GALL
CONNECTED	WORRIED	GALLING
CONNECTING	WORRIES	GALLED
CONNECTION	WORRYING	GALLEY
CONNECTIONS	WORRYINGLY	GALLERY

- **Normalisation:** heuristics to conflate variants due to spelling, hyphenation, spaces, etc.
e.g. USA and U.S.A. and U S A → USA
e.g. chequebook and cheque book → cheque book
e.g. chequebook and cheque book → cheque book

Worf Frequency and Term Usefulness:

The most and least frequent terms are not the most useful for retrieval



Stemming

Stopwords

Use Stop list removal to exclude “non-content” words

Usually most frequent (and least useful for retrieval)

a	always	both
about	am	being
above	among	co
across	amongst	could

- greatly reduces the size of the inverted index
- but what if we want to search for phrases that include these terms?
 - Kings of Leon
 - Let it be
 - To be or not to be
 - Flights to London

Single vs. Multi-word Terms

To aid recognition of phrases, might allow multi-word terms

- e.g. Sheffield University

e.g. bigram indexing: store each bigram as a term in index

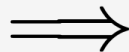
For **pease porridge in the pot** get:

pease porridge
porridge in
in the
the pot

- Problem: number of bigrams is v.large c.f. number of words
 - leads to a huge increase in size of the index
- Alternative: identify multi-word phrases during retrieval
 - Positional indexes, storing position terms in documents, can help
 - use to compute if occurrences of search terms in document are adjacent / close / far apart

Positional indexes:

Doc	Text
1	Pease porridge hot, pease porridge cold
2	Pease porridge in the pot
3	Nine days old
4	Some like it hot, some like it cold
5	Some like it in the pot
6	Nine days old



Num	Token	Docs
1	cold	1:(6), 4:(8)
2	days	3:(2), 6:(2)
3	hot	1:(3), 4:(4)
4	in	2:(3), 5:(4)
5	it	4:(3, 7), 5:(3)
6	like	4:(2, 6), 5:(2)
7	nine	3:(1), 6:(1)
8	old	3:(3), 6:(3)
9	pease	1:(1, 4), 2:(1)
10	porridge	1:(2, 5), 2:(2)
11	pot	2:(5), 5:(6)
12	some	4:(1, 5), 5:(1)
13	the	2:(4), 5:(5)

Term Weighting

What do we use for the inverted index?

- binary weights - 0/1: whether or not term is present in document
 - But documents with multiple occurrences of query keyword may be more relevant
- Frequency of term in document: like the examples we have seen
 - But what if the term is also frequent in collection?
 - Common terms: not very useful for discriminating relevant documents
- Frequency in document vs in collection: weight terms highly if

- They are frequent in relevant documents . . . but
- They are infrequent in collection as a whole

Key concepts:		
Document collection	D	collection(set) of documents
Size of collection	D	Total number of documents in collection
Term freq	$tf_{w,d}$	Number of times w occurs in document d
Collection freq	cf_w	number of times w occurs in collection
document freq	df_w	number of documents containing w

The informativeness of terms:

- Idea that less common terms are more useful to finding relevant docs:
 - i.e. these terms are more informative
- Is this idea best addressed using document frequency or collection frequency?
 - Consider following counts (from New York Times data, |D| = 10000):

Word	cf_w	df_w
insurance	10440	3997
try	10422	8760

- term insurance semantically focussed, term try very general
 - document frequency reflects this difference
 - collection frequency fails to distinguish them (i.e. very similar counts)

Informativeness is inversely related to (document) frequency

- less common terms are more useful to finding relevant documents more common terms are less useful to finding relevant documents
- Compute metric such as: $\frac{|D|}{df_w}$
 - Value reduces as df_w gets larger, tending to 1 as df_w approaches |D|
 - e.g. $\frac{10000}{3997} = 2.5(\text{insurance})$ $\frac{10000}{8760} = 1.14(\text{try})$
 - Value very large for small df_w — over-weights such cases
 - e.g. $\frac{10000}{350} = 28.6(\text{mischie f})$
- To moderate this, take log: **Inverse document frequency (idf)**
 - $idf_{w,D} = \log \frac{|D|}{df_w}$ e.g. $\log \frac{10000}{3997} = 0.398(\text{insurance})$
 $\log \frac{10000}{8760} = 0.057(\text{try})$ $\log \frac{10000}{350} = 1.456(\text{mischie f})$

BUT Not all terms describe a document equally well

Putting it all together: tf.idf

- Terms which are frequent in a document are better:
 - $tf_{w,d} = freq_{w,d}$
- Terms that are rare in the document collection are better:
 - $idf_{w,D} = \log \frac{|D|}{df_w}$
- Combine the two to give tf.idf term weighting:
 - $tf.idf_{w,d,D} = tf_{w,d} \cdot idf_{w,D}$

Most commonly used method for term weighting.

- Used in other fields too (e.g. summarisation)

TF.IDF EXAMPLE

tf.idf example:

Term	<i>tf</i>	<i>df</i>	$ D $	<i>idf</i>	<i>tf.idf</i>
the	312	28,799	30,000	0.018	5.54
in	179	26,452	30,000	0.055	9.78
general	136	179	30,000	2.224	302.50
fact	131	231	30,000	2.114	276.87
explosives	63	98	30,000	2.486	156.61
nations	45	142	30,000	2.325	104.62
haven	37	227	30,000	2.121	78.48

For term **the**:

$$idf(the) = \log_{10}\left(\frac{30,000}{28,799}\right) = 0.018$$

$$tf.idf(the) = 312 \cdot 0.018 = 5.54$$

Putting things together: Vector Space Model + tf.idf term weighting + cosine similarity

- tf.idf values for words in two documents D_1 and D_2 , and in a query Q "hunter gatherer Scandinavia":

	Q	D ₁	D ₂	
hunter	19.2	56.4	112.2	
gatherer	34.5	122.4	0	
Scandinavia	13.9	0	30.9	
30,000	0	457.2	0	
years	0	12.4	0	
BC	0	200.2	0	
prehistoric	0	45.3	0	
deer	0	0	23.6	
rifle	0	0	452.2	
Mesolithic	0	344.2	0	
$\sqrt{\sum_{i=1}^n x_i^2}$	41.9	622.9	467.5	(i.e. length of vector)

$$\text{sim}(\vec{q}, \vec{d}) = \cos(\vec{q}, \vec{d}) = \frac{\sum_{i=1}^n q_i d_i}{\sqrt{\sum_{i=1}^n q_i^2} \sqrt{\sum_{i=1}^n d_i^2}}$$

$$\begin{aligned} \cos(Q, D_1) &= \frac{(19.2 * 56.4) + (34.5 * 122.4) + \dots + (0 * 0) + (0 * 344.2)}{41.9 * 622.9} \\ &= \frac{5305.68}{26071.72} \\ &= 0.20 \\ \cos(Q, D_2) &= \frac{(19.2 * 112.2) + (34.5 * 0) + \dots + (0.0 * 452.2) + (0.0 * 0.0)}{41.9 * 467.5} \\ &= \frac{2583.8}{19570.0} \\ &= 0.13 \end{aligned}$$

- so document D 1 is more similar to Q than D 2

Web Search Ranking

Web docs contain info beyond their mere “textual content”

- state-of-the-art web search engines, like Google, exploit this
- achieve much more effective retrieval than could without it

HTML contains clues that some terms are more important

- e.g. terms in regions marked as title or headings
- e.g. terms emphasised by formatting: bold / bigger / colour
 - can use clever term weighting schemes, that add weight to such terms

Link text — commonly provide description of target doc

- often a better description than doc provides of itself
 - e.g. “Hey, here’s a great intro to calculus for beginners – check it out!”
- Google treats link text as part of target doc
- Link structure of web more broadly
 - if page A points to page B, implies B is worth looking at
 - can be used as a measure of authority / quality

Exploiting Link Structure: the PageRank Algorithm

Key method to exploit link structure of web: PageRank algorithm

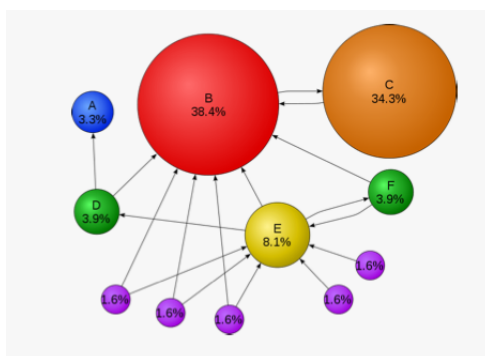
- named after its inventor: Larry Page (co-founder of Google)
- assigns a score to each page on web: its PageRank score
 - can be seen to represent the page's authority (or quality)

PageRank algorithm — key idea:

- link from page A to page B confers authority on B
- how much authority is conferred depends on:
 - the authority (PageRank score) of A, and its number of out-going links
i.e. A's authority is shared out amongst its out-going links
 - note that this measure is recursively defined
score of any page depends on score of every other page

PageRank scores have an alternative interpretation:

- probability that a random surfer will visit that page
i.e. one who starts at a random page, clicks randomly-chosen links forward, then (getting bored) jumps to a new random page, and so on . . .



During retrieval, rank score of doc d is a weighted combination of:

- its PageRank score: a measure of its authority
- its IR-Score: how well d matches the query q, based on
 - Vector Space model, TF.IDF, up-weighting of important terms, etc

Evaluation

There are various retrieval models/algorithms/IR systems, How determine which is the best?

What is the best component/technique for:

- Ranking? (cosine, dot-product, . . .)
- Term selection? (stopword removal, stemming, . . .)
- Term weighting? (binary, TF, TF.IDF, . . .)

How far down the ranked list will a user need to look to find some/all relevant items?

Evaluation of effectiveness in relation to the relevance of the documents retrieved

Relevance is judged in a binary way, even if it is in fact a continuous judgement

- Impossible when the task is to rank thousands or millions of options: too subjective, too difficult

Other factors could also be evaluated:

- User effort/ease of use
- Response time
- Form of presentation

In IR research/development scenarios, one cannot afford humans looking at results of every system/variant of system

Instead, performance measured/compared using a pre-created benchmarking corpus, a.k.a. gold-standard dataset, which provides:

- a standard set of documents, and queries
- a list of documents judged relevant for each query, by human subjects
- relevance scores, usually treated as binary

Example: TREC IR evaluation corpora (<http://trec.nist.gov/>) TREC has run annually since 1991

Metrics

AIM: 1. get as much good stuff as possible; 2. get as little junk as possible

The two aspects of this aim are addressed by two separate measures — recall and precision

	relevant	non-relevant	total
Retrieved	A	B	A+B
Not retrived	C	D	C+D
Total	A+C	B+D	A+B+C+D

recall: $\frac{A}{A+C}$ = proportion of relevant documents returned

precision: $\frac{A}{A+B}$ = proportion of relevant documents retrieved documents that are relevant

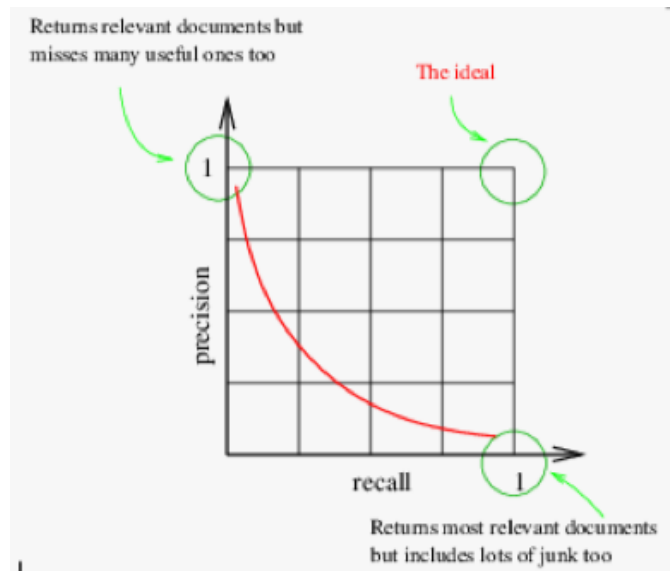
Both range [0~1]

Precision and Recall address the relation between the retrieved and relevant sets of documents

Various situations that arise can be pictorially represented in these terms

There is always a trade-off between precision and recall

For IR: as more results are considered down the list, precision generally drops, while recall generally increases



F measure (also called F1): combines precision and recall into a single figure, gives equal weight to both: $F = \frac{2PR}{P+R}$

F is a harmonic mean: penalises low performance in one value more than arithmetic mean:

	values	mean	F
e.g.	P=0.5, R=0.5	0.5	0.5
	P=0.1, R=0.9	0.5	0.18
Previous example:			
	R	P	F
System 1	.57	.64	0.603
System 2	.43	.8	0.559

Related measure F_β :

- allows user to determine relative importance of P vs. R, by varying β
- F1 is a special case of F_β (where $\beta = 1$)

Precision at a cutoff

Measures how well a method ranks relevant documents before non-relevant documents

- E.g. there are **5 relevant documents = d1,d2,d3,d4,d5** – compute precision at top 5

	System 1	System 2	System 3
	d1: ✓	d10: ✗	d6: ✗
	d2: ✓	d9: ✗	d1: ✓
	d3: ✓	d8: ✗	d2: ✓
	d4: ✓	d7: ✗	d10: ✗
rank 5:	d5: ✓	d6: ✗	d9: ✗
	d6: ✗	d1: ✓	d3: ✓
	d7: ✗	d2: ✓	d5: ✓
	d8: ✗	d3: ✓	d4: ✓
	d9: ✗	d4: ✓	d7: ✗
rank 10:	d10: ✗	d5: ✓	d8: ✗
precision at rank 5:	1.0	0.0	0.4
precision at rank 10:	0.5	0.5	0.5

Note precision at top 5 for System 1: inner order of relevant documents doesn't matter as long as they are all relevant

	System 1	System 2	System 3
	d5: ✓	d10: ✗	d6: ✗
	d4: ✓	d9: ✗	d1: ✓
	d3: ✓	d8: ✗	d2: ✓
	d1: ✓	d7: ✗	d10: ✗
rank 5:	d2: ✓	d6: ✗	d9: ✗
	d6: ✗	d1: ✓	d3: ✓
	d7: ✗	d2: ✓	d5: ✓
	d8: ✗	d3: ✓	d4: ✓
	d9: ✗	d4: ✓	d7: ✗
rank 10:	d10: ✗	d5: ✓	d8: ✗
precision at rank 5:	1.0	0.0	0.4
precision at rank 10:	0.5	0.5	0.5

Average Precision

- Aggregates many precision numbers into one evaluation figure
- Precision computed for each point a relevant document is found, and figures averaged

System 1		System 2		System 3	
d1:	✓ (1/1)	d10:	×	d6:	×
d2:	✓ (2/2)	d9:	×	d1:	✓ (1/2)
d3:	✓ (3/3)	d8:	×	d2:	✓ (2/3)
d4:	✓ (4/4)	d7:	×	d10:	×
d5:	✓ (5/5)	d6:	×	d9:	×
d6:	×	d1:	✓ (1/6)	d3:	✓ (3/6)
d7:	×	d2:	✓ (2/7)	d5:	✓ (4/7)
d8:	×	d3:	✓ (3/8)	d4:	✓ (5/8)
d9:	×	d4:	✓ (4/9)	d7:	×
d10:	×	d5:	✓ (5/10)	d8:	×
precision at rank 5:		1.0	0.0	0.4	
precision at rank 10:		0.5	0.5	0.5	
avg. prec:		1.0	0.354	0.573	