L01: Search Engine (SE)

Names for different retrieval

Information (IR)	Document (DR)	Text (TR)

Specific SE

Vertical	One type of data (Job search, news search)
Site	Just one site
Custom	Narrow search to small set of websites
Enterprise	Corporate intranet w/ collections, metadata, roles, security

Federated search organized: Meta search "janky" option

Federated search (union)	Meta search (calling bots)
Each node full-function SE	It itself is not a SE
Agreement obtained	No agreement: May blocked
Standard query/result repr.	Nonstandard and parsing
SEs collaborate	SEs do their own thing

Difficulties of IR

Unstructured text / media	Size(cloud)	Semantics	Diversity> Personalize

Cloud computing advantages (combat size issue)

Unlimited scaling	Many replicas and shards

Indexing by professionals

High cost	Inconsistent	Inefficient

Timeline of IR generations 0 to 3

1960: exact bool	1993: stat. crawler	1997: link analysis	2001: advanced

L02: Business model ($$$)

Software Licensing	Sell the software for intranet (and/or internet) search
Search Service	Outsourced content portals, monetize by ads (e.g. Yahoo)
ASP/Cloud	Do the HW/SW/Index dirty work for paying clients
Advertisement	Keyword-based ads / organic results. >> popup&banners
Charge end users	End user refuses to pay unless unique valuable info.
Charge websites	Charge by queries, pay for indexing, pay for ranking

Notes

Keyword ads	AdWords (buy keywords) AdSense (analyze page content)
(Content)Portal	SEs used to need portals: now SE=portal, need ad channels
SEM(Marketing)	SEM Company help get inclusion, find keywords rank high
SEP(Placement)	Pay-by-Impression (Cost Per Mille=1000) / Pay-per-Click
SEO(Optimization)	Make webpage rank high in algorithm
Trademark KW	Potential legal problems such as lawsuits

Retrieval models

Performance metrics

Precision: % of relevancy in retrieval (Noise in retrieval)	Recall: % of relevant results retrieved (Completeness)

Model contains details

Document repr.	Query repr.	Retrieval function

Current class of models

Boolean model	Vector space (stat)	Probabilistic

L03: Boolean model

Document Repr.	Set of keywords (bag of words BOW)
Query Repr.	Boolean expression of keywords (AND, OR, NOT, brackets)
Retrieval fn.	Satisfy the Boolean query? YES/NO
Pros	Simple, controllable, efficient, extendable to include ranking
Cons	Very rigid, no complex requests, AND/OR interpretation No control # docs, no ranking, no auto relevance feedback

L03+L04: Statistical models (MORE LIKE OR)

Document Repr.	Set of keywords WITH statistical information DI= < Text I.O: database 0.5: information 0.2 > Weight roughly indicates important of word Weighted KW as row vector, 0 for missing
Query Repr.	Keywords with optional weights. NO Boolean Query keywords (optional weights) as vector Without weights: 1 if term present, else 0
Retrieval fn.	Similarity between query and documents
Challenges	Word importance (within doc, entire collectn.) Degree of similarity b/w document & query (WWW) What is collection? Links, doc structure, format info Zero weight causes math error in some algo

L03: TFxIDF: Word importance

(within document TF & entire collection IDF)

Word importance (intra-document): Term Frequency (TF)	**Principle** repeat = more important **Issue**: Unbounded due to diff. doc size **Fix**: Normalize by division (TFmax, # of (unique) words) **Mod**: range from 0.5 to 1, favor small tf, avoid zero weight
Word importance (inter-document): Inverse Document Frequency (IDF)	**Principle** rare = discriminant, formula: log(N/dfi) **Issue**: N=0 cause **Fix**: log(1+N/dfi) ranges from log(2) to log(N+1) avoid zero
BM25 weight (OKAPI)	Average doc length affects weighting Pre-computation is expensive
TFxIDF	TF and IDF anticorrelated, TFxIDF max when both medium values

L04: Vector Space Model (similarity b/w doc and query)

Similarity functions

Inner Product (similarity)	Formula: term-wise multiply and then sum unbounded, favors long documents
Cosine similarity	Fmla: (Inner Product) / DocLength (DL) / QueryLength (QL) (cosine of angle)
Jaccard Coeff	Formula: (IP) / (DL^2 + QL^2- IP) Intersection size divided by union size
Dice Coeff	Formula: 2 IP / (DL^2 + QL^2) Does not satisfy triangle inequality!

Operations on vectors

Relevance feedback: Q+DI or Q-DI to add/subtract good/bad terms

Flexible use of similarity on different vector pairs

Doc-query: SE (typical)	Doc-doc: doc clustering	Query-query: Query suggestion

Document centroid: reduce similarity pairs checking doc similarity.

First find the centroid, then calculate pairs similarity centroid<->doc

Comparison between similarity functions

unmatched terms matter much less in doc-query than doc-doc

	Euclidean	Cosine	Inner	Jaccard
Doc-Doc	OK	OK	Fair	Fair+
Doc-Query	Bad	OK	OK	OK

Issues

Term Independence Assumption	Hardly holds true in real life
Synonyms (fix by stemming)	Causes false negative matches
Unbalanced properly (fix by penalizing)	Docs without including all terms but talks a lot about it gets included
OR-like nature	First Boolean, then ran via VSM

L05: Inverted Files

Green: Optional to be stored

Document vectors/matrix:

sparse, not stored directly. Use a word list instead.

Indexing: speedup slow algorithms using **inverted file**.

Inverted file (document index)

Data struct	Hash file, B-tree, tries, etc: Key-value: as str
Data stored	Key: term, Value: list len(DF) postings (doc, TF)

Forward index

Data stored	Key: doc, Value: list of terms & location, TFmax

Boolean query

Get all postings and then perform set operations.

Optimize: start w/ AND on shortest lists: intermediate results small.

Advantages of inverted file (document index)

Filter documents that contain query term for fast retrieval	Flexible structure: Store "postings" with info (doc with terms)	Cmplx retrieval fn. w/ word locations (phrase & proximity)

Disadvantages of inverted file (document index)

Large storage overhead: (50% - 150% - 300%)	High maintenance costs on updates, insertions, deletions	Processing cost increases with # of Boolean operators

Extensions on Inverted Indexes

Adjacency	Phrase, n words apart, same sentence. Require storing keyword locations & document components
Term truncation (suffix / prefix)	Searching for "comput*" (suffix) / "*symmetry" (prefix, hard) Require storing inverted index as tree / B+-Tree, no hash file

Overhead in inverted indexing on addition

Worst: words unique, doc n words, insertion update n postings lists.

Unsorted posting list: Fast append, slow retrieval

Sorted postings list: Slow insertion, fast retrieval

Insertion overhead: Big issue for news and WWW, not for library

Speed improvement: batch insertion (>1 doc has the same word)

Overhead in inverted indexing on deletion and update

Update is just deletion and insertion. Deletion is expensive.

To reduce the cost, keep table of deleted docs for lazy deletion.

Table of del. docs	Ignore del. docs	Clean inverted file

Scalability and speed

Indexed keywords grow slowly, but length of postings lists increase linearly. As such, partition a collection to be index/searched in parallel by different servers.

L06: Extended Boolean model

Issue of Boolean->VSM:

AND: Boolean too restrictive	OR: Same as pure VSM

Soft logical operators

AND: big penalty for 1 term $sim(q_{and}, d_j)$ $= 1 - [((1-x)^2 + (1-y)^2)/2]^{1/2}$ Complement vec len from (1,1)	OR: small bonus for both $sim(q_{or}, d_j)$ $= [(x^2 + y^2)/2]^{1/2}$ Vector length from (0,0)

Replace above formula 2 to p for p-norm model

p=1: similar to inner product in VSM; p=inf: fuzzy logic model

L07: Link-based ranking

Existing issues with term-based methods

Keyword spamming	Doc content insufficient	Agnostic to user stats

Link-based ranking assumption and principle

Linked pages assumed similar	Return w/o exact terms

HyPursuit (MIT, 1996)

(The formulas are ad-hoc. Not directly usable in ranking, but for clustering to enhance retrieval speed and quality)

Direct Path	$S_{ij}^{spl} = \frac{1}{2^{spl_{i \to j}}} + \frac{1}{2^{spl_{j \to i}}}$

Common ancestors	$S_{ij}^{anc} = \sum_{x \in X} \frac{1}{2^{spl_{x \to i,no\ j} + spl_{x \to j,no\ i}}}$
Common descendants	$S_{ij}^{dsc} = \sum_{x \in X} \frac{1}{2^{spl_{i \to x,no\ j} + spl_{j \to x,no\ i}}}$

Somehow (ignored) "combine" metrics w/ term-based similarity.

WWW Index and Search Engine (WISE) at HKUST (1995)

Page score is the weighted sum of:

its term-based score	scores inherited from parents

With $\beta << \alpha$: score = α(exact matches) + β(match from parent)

Most-cited: score = # of query terms found in pages pointing at it (frequency information is ignored)

Problems

Related != similar	Void assumption of link-based ranking
Diff b/w web & traditional docs	Links find more results, we don't need! Links (abused!) promotes page rank
Quality&Authority (Application aspect)	Blogpost "I dream study at (link)HKUST" In web, doc have diff. quality/authority
Technical aspect	No theoretical/systematic way to set fmla (WISE > HyPursuit)

PageRank (Google, Stanford U) (NOTE: Query independent!)

(Academic citation: More cited = important page, spam-resistant)

Equals random surfer model: chance (1-d):type URL, d:click link.

Uses additional data to its advantage:

link text	Location info	Visual info	Social signals

$$\text{PageRank}(A) = (1-d) + d \sum_{P \in \text{direct parent}} \frac{\text{PageRank}(P)}{\#\text{ links from}(P)}$$ d = damping factor

First mark all as 1, then iterative algorithm until it converges.

Sync iteration	A,B,C = PR(A), PR(B), PR(C)
Async iteration (order dependent)	A = PR(A); B = PR(B); C = PR(C) PR considers value of A B C

Expensive! Parallelized, updated lazily (Google Dance) & inexactly

Issues

Favor big websites (discovery issue)	Linked != relevant (Void assumption)	Rank sink: point each other, no d

Anchor text ranking

If see (Computer Science Department)(http://www.cse.ust.hk), Proceed to associate the URL with the human readable name tag.

SEO: Link boosting

Split content	Large site big PageRank (favor big websites)
Limit outlinks	PageRank leak. (w/ many internal outlinks)
Avoid sinks	Never have user stuck (purge!), +graceful exit

Spider issues

Web server hangs	Bad last-modified	Redirect. password
Dup. webpages	Different lang. ver	Dynamic (JS)

L08: HITS (Hyperlink-Induced Topic Search)

(Highly efficient compared to tfxidf: simple combination of scores)

Parents have high hub weight	High authority: Quality info
High hub: link to high quality	Children have high authority

Each page authority weight and hub weight recursive calculate:

Authority of p is the sum of hub weights of all q where $q \to p$
Hub of p is the sum of authority weights of all q where $p \to q$

If authority vector is X, hub vector is Y (independent to initial val):

Authority: $X_i = A^T Y_0$	Hub: $Y_i = A X_0$
Converge to A^tA	Converge to AA^t

Reasons for non-convergence

Dangling nodes: to infinity	Rank sinks: Oscillating values

Normalize each iter (stop overflow, not affect final ranking)

Div largest element	L1: div sum	L2: div magnitude

CLEVER architecture

Basic search	Parent and child	HITS then re-rank

Issues with HITS

Children reinforce each other	Multi-topic page

L09: Performance Evaluation

Reason for motivation

Make comparison	Test for deviation	Fine tune query
Cost-benefit analysis (compare to manual)		Effect of change

Evaluation modes

Explicit	Human judge relevance* offline. Issues: guess need and intent, human error, inefficient, expensive, inconsistent (auto filtering?) *(continuous, subjective, situational, temporal)
Behavioral	Clicks and timestamps. Online and Realtime Metric: Average Rank of User Clicks (ARUC)

Confusion matrix

Retrieved irrelevant: FP	Not retrieved, irrelevant: TN
Retrieved relevant: TP	Not retrieved, relevant: FN

Recall = TP / (TP+FN) (div0 if no relevant).

Precision = TP / (TP+FP) (div0 if no document retrieved)

Recall and precision difficult to have both high (graph)

Fallout: FP / (FP+TN) (prob of non-relevant, inv. of recall, no div0)

F1-measure: 2PR/(P+R)

Total number of relevant items

Hard to know. Sample collection, or apply different retrieval algo

Recall-precision graph

Upper right hand corner = better. Compare system by average.

For a certain recall level, precision could be missing. Interpolate!

Precisions at the "standard 11 recall levels": 0, 0.1, ... 0.9, 1.0. evaluate

How to also consider ranking?

Moving recall, Top-k precision, avg. precision (=Area Under Curve)

Discounted Cumulative Gain (DCG) (wrongly low rank = penalize)

$$DCG_p = \sum_{i=1}^{p} \frac{rel_i}{\log_2(i+1)}$$

Mean Reciprocal Rank: Reciprocal of first relevant document hit.

Subjective relevance measure

Novelty ratio	Coverage ratio	Sought recall

Other factors

User effort (UI, Query)	Response time (trade)	Form (UI) of Presentation	Collection coverage

L10: Benchmarking

Standard docs and queries, relevant docs for each query

SMART collection	Text Retrieval Conference

Issue: Benchmark-specific, resource-consuming, language

L11: Text processing

Information Retrieval workflow (typically use term IDs):

Tokenize*	Stem*	X stopwords	Indexing

*NLP workflow adds to after the tokenize:

Lemmatize	Part-of-speech	NER (entity recog)

Stopword uses: avoid index common words, Must consistent!

Stemming uses: unify variations, shrink index, enhance recall

Affix removal	Successor variety	Table lookup	N-gram	Statistical (corpus)

Famous algorithm: Porter's algorithm 1960s (language dependent)

(NOT linguistically accurate, for improve IR only, fries -> fr)

*If do not stemming, expand query terms (computer += computation), but time consuming due to many terms

L12: Relevance feedback (user lazy and dumb!)

User lazy and dumb for query	Ambiguity (disambiguation?)

Method: Query Expansion/Reformulation/Modification

Manual User lazy and dumb	**Automatic** relevance feedback

Method: Implicit vs Explicit feedback

Explicit User lazy and dumb	**Implicit** relevance feedback

Implicit relevance feedback includes their actions (clicks)

Relevant: reinforce vector	Nonrelevant: weaken vector

(Not just adjust query term weight but can adjust document too!)

Big business. Correctness > relevance: few but unmissable results. Specific queries, no links, not webpages, security, flexible scoring.

L13: Personalization

Cause: Not one search fits all. Capture user interests (L12), then re-rank. As always, implicit preferred, as it is easy collectable despite lazy and dumb user, reflective of real usage.

Important metrics

Click-through rate	Browsed documents

Absolute feedback: Relevant or irrelevant. Issue:

Click != relevant	Click paradox: user trust SE, SE trust user

Relative feedback, more relevant or less relevant in comparison

Not clicked == not relevant	More reliable, 80% correct

How to measure?

Eye tracking: 200-300ms fixation, 40-50ms saccades pupil dilation?	Clickthrough analysis: Record rank & clicked pages (assume read sequentially, click relevant, skip irrelevant)

Clickthrough analysis logic

Best Performance (correlation 80%-90%)	Average Performance (correlation 65% to 75%)
(1) Click > Skipped Above	(3) Click > Earlier Click
(2) Last click > Skip Above	(5) Click > No Click Next
(4) Click > Skip Previous	

How to rerank it

Add personal weight vector	Query reformulation

L14: Index Term Selection

Indexing is pre-requisite to high quality search output!

Zipfs law (useful for SE, dictionary-free stop words)

Due to principle of least effort, frequency f of a word is inverse of the rank r in usage popularity. Frequent words are short and account 60% of all word usage. Meanings m correlate to sqrt(f) Content words are words which occur at most 24 times in the Corpus. These tend to clump together and for number of times F that two content words appear in interval size I (I = 1, 2, 3, etc.), F correlate I^-p where p from 1 to 1.3

Index term selection

Too few is inaccurate, too much is noise. Want only the ones with high discriminate ability. DV (discrimination value) anticorrelate with idf. As usual, calculate every pair is expensive. Calculate document centroid for ease of calculation.

Complete Term Selection Algorithm

1. Identify all words in the documents
2. Select a similarity measure (say, inner product)
3. Set all term weights to 1
4. Candidates = all terms
5. Compute dv's for all Candidates
6. Select the term with the highest dv
7. Remove the term from Candidates
8. Repeat 5-7 until the desired number of terms are selected

L15: Discovering Phrases and Correlated Terms

N-grams: number of unique n-grams is large!

Improve recall but hurt precision

Some N-grams are meaningless, so use filtering:

Stop words	Frequency	Grammar

To detect collocations and co-occurences (interchangeable term):

High frequency of bigrams	Grammar pattern matches

pointwise mutual information (PMI):

normalize co-occurrence frequency:

$$I(x,y) = \log_2 \frac{p(x,y)}{p(x)p(y)}$$
$$= \log_2 \frac{N\text{freq}(x,y)}{\text{freq}(x)\text{freq}(y)}$$ N is number of words

0 when uncorrelated, +ve is correlated, -ve is anti-correlated

Applications: reveal real word knowledge, query suggestion, identify page topic, page summarization, page ranking, find spam.

L16: Corporate SE (mostly ignored)