HKUST COMP 4321 cheatsheet

search engine (SE)

- names: information retrieval (IR), document retrieval (DR), text retrieval (TR)
- domain-specific SE: domain/vertical, site, custom, enterprise (more metadata), embedded
- federated & meta SE: multiple search backends
  - federated: collaborated, parent SE is a SE itself
  - meta: opposite of federated
- why is IR difficult: many unstructured text, semantics, personalization
- cloud computing: unlimited scaling, many replicas and shards
- indexing by professionals: inconsistent, inefficient
- generations: 0: 1960 (exact, bool); 1: 1993 (+statistical); 2: 1997 (+links); 3: 2001 (+advanced features)

business model

- sell software
- search service used by other SEs
- custom search (ASP)
- keyword-based ads
- others: charge user, websites, for indexing, for ranking
  - unworkable, unless high quality
- keyword ads: AdWords (search), AdSense (page content)
- portal: before, SEs need portal traffic; now, big SEs become portal themselves
- SEM/SEO/SEP: promote ranking
- ad pricing: per impress (PPM), per click (PPC); CPM: cost per mille (thousand)
- selling trademarked keywords: legal trouble

TF×IDF

- model specs: document format, query format, retrieve function
- models: bool, (statistical) vector, probabilistic
  - bool model
    - document: list of keywords (no count)

    - retrieve function: AND, OR, NOT; check satisfy
    - good: controllable, efficient, simple
    - (see extended bool model)
    - bad: AND/OR meaning, complex queries, control of # results, ranking, relevance feedback, rigid
- precision, recall
- vector model:
    - document: list of weighted keywords
    - retrieve function: no boolean, similarity-based
    - problems: determine weight, doc—doc & doc—query similarity, metadata in content (links)
    - term frequency (TF)
        - need normalize
        - variants: $0.5 + 0.5\mathrm{tf}_{\mathrm{norm}}$
    - inverse document frequency (IDF): $\log_2(N/\mathrm{df})$
        - variants: $\log_2(1 + N/\mathrm{df})$
    - BM25 (OKAPI):

$$w = \left(\ln \frac{N - \mathrm{df} + .5}{\mathrm{df} + .5}\right) \frac{(k+1)\mathrm{tf}}{\left((1-b)k + b\frac{\mathrm{dlen}}{\mathrm{dlen}_{\mathrm{avg}}}\right) + \mathrm{tf}} \qquad k = 2, b = .75$$

        - precompute expensive, dynamic okay
    - TF×IDF: anti-correlated; highest when both are medium

vector space model (VSM)

- document: list of weighted keywords, reinterpret as a row vector; fill in 0 for nonexistent keywords
- documents: multiple row vectors as matrix
- interpretation: *n*-dimensional vector space, where *n* is the number of terms in the vocabulary
- similarity measure: between doc—doc, doc—query, query—query; ranking & truncation, modify query by relevance feedback
    - inner product: unbounded, favors long documents;

$$\mathrm{inner}(\vec{a}, \vec{b}) = \sum_{k=1}^{n} a_k b_k$$

    - length:

$$\mathrm{len}(\vec{a}) = \sqrt{\mathrm{inner}(\vec{a}, \vec{a})}$$

    - cosine similarity:

$$\frac{\text{inner}(\vec{a}, \vec{b})}{\text{len}(\vec{a})\,\text{len}(\vec{b})}$$

    - Jaccard coefficient: Paul Jaccard in 1901;

$$\frac{\text{inner}(\vec{a}, \vec{b})}{\text{len}^2(\vec{a}) + \text{len}^2(\vec{b}) - \text{inner}(\vec{a}, \vec{b})}$$

    - Dice coefficient: 0 to 1, no triangle inequality;

$$\frac{2\,\text{inner}(\vec{a}, \vec{b})}{\text{len}^2(\vec{a}) + \text{len}^2(\vec{b})}$$

    - binary versions: replace true/false with 1/0; the above formulas can be re-described in terms of number of 1s, union, and intersection
- query: treat the query similar to a document and make a row vector
    - similar to OR in bool model
- vector operations: implement relevance feedback
- relevance feedback: good terms (reinforce), good terms missed (add), bad terms (subtract)
    - bool model: ill-defined, queries become complex
- similarity uses: ranking, clustering, query expansion/suggestion
- clustering: document centroid
- problems: term independence, synonyms
    - similarity measure choice: unmatched terms matter much less in doc—query than doc—doc, table below
    - unbalanced: <x, x> and <x, y> can be equally similar to a query; reward balanced and penalize unbalanced, or multiply similarity by $|D \cap Q|/|D \cup Q|$
- add bool model: bool model as filter, then VSM as ranker

	doc—doc	**doc—query**
Euclidean	okay	bad
cosine	okay	okay
inner	mediocre	okay
Jaccard	mediocre	okay

inverted files

- document vectors/matrix: sparse, not stored directly
- indexing: speedup slow algorithms using inverted file
  - data structures: hash file, B-tree, tries (singular trie), etc.
  - key -> value: both stored as strings
  - inverted index: term, DF (optional, can be separate) -> (doc, TF) (posting in postings list), positions (optional)
  - forward index: doc -> list of terms, TF max (optional)
  - separate file: DF, N
- represent words and docs with integer IDs
- bool query: treated as bool terms connected by bool operators
  - optimization: start with AND terms that return smallest postings lists; smallest intermediates
- good: fast retrieval, flexible structure (extra data), support complex operations/extensions
  - phrase/proximity (e.g. immediate after, 3 words apart, same sentence): positions (character/word/sentence/paragraph position), document structure (e.g. sentence breaks), and extended retrieval function
  - term truncation (e.g. `comput*`): tree easy, b+-tree harder, hash file infeasible
  - prefix truncation (e.g. `*symmetry`): tree, b+-tree, hash file all difficult
- bad: storage (50%~150%~300%), higher create/update/delete cost, cost increase with # bool operation; for relatively static env
  - storage overhead (in %) = index size / text size

index overhead

- algorithm
  - for each term Q_i in Q (complexity: all terms)
    - for each doc D (complexity: all docs)
      - lookup Q_i in D's postings list (complexity: half of the postings)
      - $\text{score}(Q, D) \mathrel{+}= \text{partial score}(Q_i, D)$
- inverted overhead: consider n unique terms in a doc, sorted or unsorted postings list
- overhead in practice: design requirements (library vs. newspapers/WWW), disk and memory size, batch/online update
- forward index: doc -> list of terms, TF max (optional)
  - support delete

- update = delete + create
- decrease delete cost: mark as deleted, and periodically actual delete (maintenance)
- typical indexes: map: term -> ID, URL -> ID; inverted: word -> (page ID, positions); forward: page ID -> terms; properties: page ID -> title, URL, last mod, size, etc.
- link-based index (adjacency list): child ID -> parent IDs
- batch/bulk insertion: index new documents starting with a blank index, then merge with master index
  - blank index is cheap and small, maybe can be in memory
  - batch size: too small: less chance of same term in many docs; too big: insert into blank index expensive
  - new documents are delayed

scalability

- as # docs increases...
  - # terms increases increasing slowly (log)
  - postings list size increases linearly
- partition: partition documents and index separately
- search: search on all partitions; maybe can rank per partition first; how to merge

extended bool model

- bool filter -> VSM: AND: maybe too restricted; OR: same as VSM
- conjunctive: x AND y, $x \wedge y$
- disjunctive: x OR y, $x \vee y$
- extended bool model: smoothly approximates the bool operators
- normalize term weight x_D for term x in doc D: between 0 and 1, (TF × IDF)/(max TF × max IDF)
- smooth conjunctive $\wedge$:

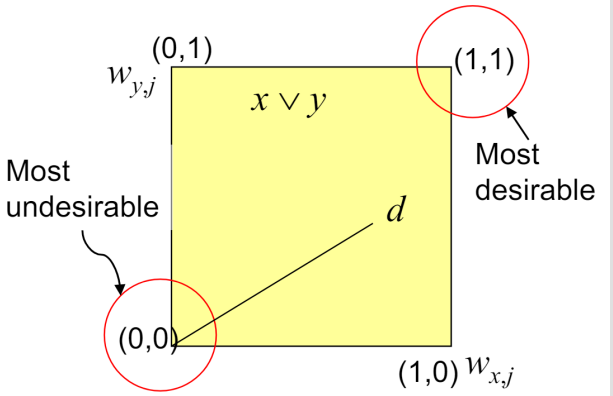
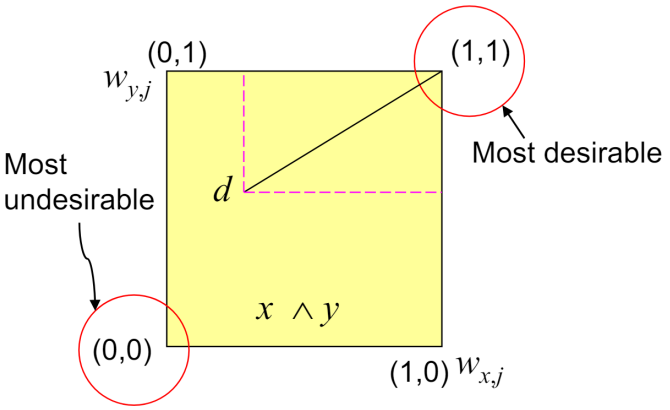
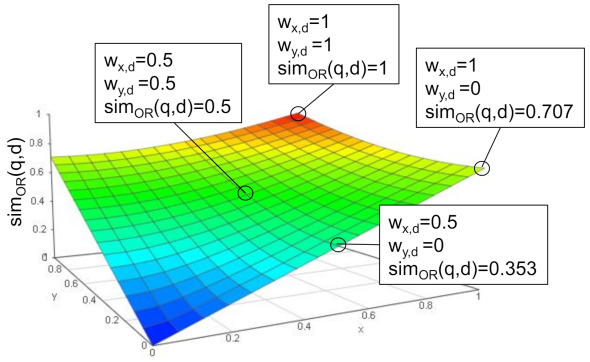
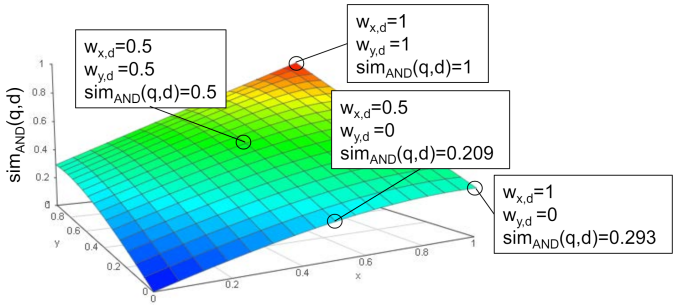
$$\text{sim}(x \vee y, D) = \left(\frac{x_D^p + y_D^p}{2} \right)^{\frac{1}{p}}$$

- smooth conjunctive $\vee$:

$$\text{sim}(x \wedge y, D) = 1 - \left(\frac{(1 - x_D)^p + (1 - y_D)^p}{2} \right)^{\frac{1}{p}}$$

- p usually 2; 1: inner product (taxicab); ∞: fuzzy logic (Chebyshev, AND = MIN, OR = MAX)

- above functions output also between 0 and 1, so can recursively evaluate
- graphical interpretation ($p = 2$): vector length divided by $\sqrt{2}$; 3D graph: concave up for OR, concave down for AND; table below
- extended bool model: $p = 2$
- p-norm model: OR: $\vee^2$, AND: $\wedge^\infty$

$x \vee y$	$x \wedge y$
	
$x \vee y$	$x \wedge y$
	

link-based ranking

- term-based only problems: pages with no mention, spamming
  - reason: doc content itself is insufficient, need: links, doc props, user stats (e.g. click-through rate), etc.
- good: linked pages are *assumed* to be related or similar, can return linked pages without the exact terms
- HyPursuit at MIT (1996)
  - shortest path length: SPL
  - direct path:

$$S_{ij}^{\text{spl}} = \frac{1}{2^{\text{spl}_{i \to j}}} + \frac{1}{2^{\text{spl}_{j \to i}}}$$

- common ancestors:

$$S_{ij}^{\text{anc}} = \sum_{x \in X} \frac{1}{2^{\text{spl}_{x \to i, \text{no } j} + \text{spl}_{x \to j, \text{no } i}}} \qquad \text{where } X \text{ are all common ancestors}$$

- common descendants:

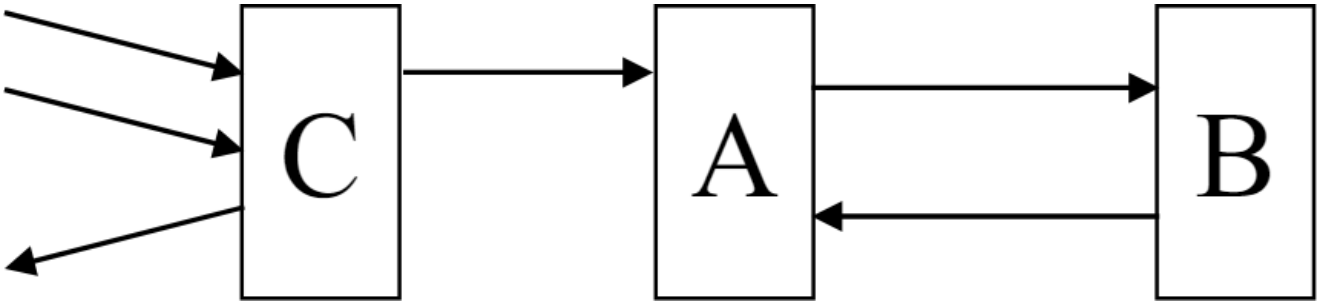
$$S_{ij}^{\text{dsc}} = \sum_{x \in X} \frac{1}{2^{\text{spl}_{i \to x, \text{no } j} + \text{spl}_{j \to x, \text{no } i}}} \qquad \text{where } X \text{ are all common descendants}$$

- link-based similarity: combine direct path, common ancestors, and common descendants; then combine with term-based similarity; combine method not matter
- link formulas are ad-hoc
- not directly usable in ranking, but for clustering to enhance retrieval speed and quality
- used to classify 195 documents from cnn.com using complete-link clustering method -> compare with CNN classes -> precision, recall

- WWW Index and Search Engine (WISE) at HKUST (1995)
  - page score is the weighted sum of its term-based score matching and scores inherited from its parents
    - $$\text{score} = \alpha(\#\text{ matches}) + \sum_{p \in \text{all direct parent links}} \beta(\#\text{ matches} \in p) \qquad \alpha << \beta$$
  - most cited equals number of query terms found in direct parents
    - term frequency is ignored
  - 1995-04-26: 2393 pages from cuhk.hk for diversity -> select 56 pages and subjectively construct 56 most relevant queries -> subjectively judge retrieved pages
- problems with HyPursuit and WISE
  - related != similar, linked may not be similar but only related
  - does not account for diff between web and traditional docs
    - links increase results, but are the results needed
    - links promote results, but are the links right
  - quality and authority differs in web; opposite for traditional docs
  - technical: no theoretical/systematic way to set the formulas (WISE > HyPursuit)
  - application: quality and authority
- Google, Standford University

- additional data: PaegRank, anchor/link text, location info, visual (e.g. font size)
- PageRank
  - similar to academic citation: a page is good when linked from many pages or from another good page; spam-resistant
  - $\text{PageRank}(A) = (1-d) + d \sum_{P \in \text{all direct parent pages}} \frac{\text{PageRank}(P)}{\#\ \text{links from}(P)}()$ damping fac
  - reiterate the above; initially make them all 1 or 1/# pages
    - sync iteration: all PR values are from last iteration
    - async iteration: PR values are updated immediately (order dependant)
  - expensive, but parallelizable as it depends on parent links only
  - no need to update if the graph updates
    - Google runs it monthly, maybe more frequently now -> Google dance
  - no need to be exact
  - bad: favors big websites but not small websites (discovery issue), general websites
    - being linked does not mean relevant
  - rank sink: no damping factor initially, group of pages that does not link to others keep increasing PR; see image below
  - random surfer model is the PR model
    - normalize PR so that PR is "probability" of a surfer at said page
    - teleport to a page with probability $1-d$ (d is damping factor)
    - go to a page from a parent is $d \cdot \text{PR}(P) / \#\ \text{links from}(P)$
    - no back or forward buttons
  - SEO: link boosting
    - split content into many pages to increase total PR
    - limit links to other websites to avoid PR leak
    - avoid sinks (no links to other websites): easily detectable
  - extensions: separate graphs for separate topics, page-specific d
- anchor/link text

    - good: more accurate/objective, describe non-indexable binary files
    - bad: linked does not exist
  - hit list: record a hit's properties: word position, font, capitalization, etc.
    - fancy hit: URL, title, link text, meta tags; plain hit
  - better than other commercial SE
  - social signal (e.g. reviews, followers)
- crawler issues: request timeout, bad last mod time, authentication, redirects, HTTPS, same page with different URLs, same content in different language, dynamic content (JS)
- compare above: link interpretation, offline/online (with query), page content dependence, query dependence



- Hyperlink-Induced Topic Search (HITS): Kleinberg 1997
  - page weights
    - authority: high when high quality info; sum of hub weights of all parent links
    - hub: high when links to high quality pages; sum of authority weights of all children links
    - recursively computed as in PR
  - connectivity/adjacency graph - example: 2 pages, A links to B;

$$A = \begin{array}{c} \\ A \\ B \end{array} \begin{array}{c} \begin{array}{cc} A & B \end{array} \\ \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \end{array}$$

  - authority vector $\vec{X}$, hub vector $\vec{Y}$: column vector, each element corresponds to 1 pagee
  - iteration 0: set all initial authority and hub weights to 1
  - power iteration: $\vec{X} \leftarrow A^T\vec{Y},\ \vec{Y} \leftarrow A\vec{X}$
    - sync: $\vec{X}, \vec{Y}$ are from the last iteration
      - $n-1$ is odd: $\vec{X}_{n-1} \leftarrow \left(A^TA\right)^{\frac{n-1}{2}}A^T\vec{Y}_0,\ \vec{Y}_{n-1} \leftarrow \left(AA^T\right)^{\frac{n-1}{2}}A\vec{X}_0$
      - n is even: $\vec{X}_n \leftarrow \left(A^TA\right)^{\frac{n}{2}}\vec{X}_0,\ \vec{Y}_n \leftarrow \left(AA^T\right)^{\frac{n}{2}}\vec{Y}_0$

    - async: update $\vec{X}$, then update $\vec{Y}$ using updated $\vec{X}$
      - $\vec{X}_n \leftarrow \left(A^T A\right)^{n-1} A^T \vec{Y}_0$, $\vec{Y}_n \leftarrow \left(AA^T\right)^n A\vec{Y}_0$
      - another form: $\vec{X}_k \leftarrow \left(A^T A\right)\vec{X}_{k-1}$, $\vec{Y}_k \leftarrow \left(AA^T\right)\vec{Y}_{k-1}$
    - authority matrix: $A^T A$, hub matrix: AA^T
  - convergence
    - principal eigenvectors of the above matrices, which is independent of the initial values
    - weights increase indefinitely; relative ranking preserved
    - do not convergence due to rank sinks (no outlinks) causing oscillation
  - normalization: avoid overflow, align with theoretical model
    - effect: no effect on final ranking
    - normalized by dividing the authority and hub vector by: largest element, L1 norm (taxicab length, random surfer model), L2 norm (Euclidean length)
    - applied after an iteration (including iteration 0)
  - diffusion effects: children of a page reinforce each other in terms of authority (prestige by association?)
- integrate PR into Google: unknown; probably ranking algorithm based on PR (periodically offline-updated) and fancy/plain hits; efficient
  - query -> keyword matcher -> matching results -> ranking (fancy/plain hits + PR) -> ranked results
- integrate HITS: CLEVER architecture by Kleinberg
  - query -> any SE -> ranked results -> re-ranking -> re-ranked results; ranked results -> graph-based information -> re-ranking
  - SE must be good enough to get good initial results
  - graph-based information
    - root set: contains query keywords; base set/topical subgraph: root set and direct parent and children of root set
      - not all pages include any queried keywords
    - HITS until convergence
    - re-rank: main results by authority, separate resource results by hub
- a page linking to multiple distinct topics/mixed hub: both algos diffuse the quality of a topic to other topics, more so for HITS, because all links are considered the same

- manipulation: a page can modify itself to change authority and hub easily; but not PageRank

performance evaluation

- why: compare, test for deviation, fine tine query, cost-benefit analysis, determine change in SE -> effect
- efficiency, effectiveness
- evaluations
  - explicit: human judges determine the relevance of ranked results given queries, offline
  - behavioral: real user behavior logged (e.g. click, timestamp), online/realtime, metrics (e.g. total number of clicks, average rank of user clicks (ARUC))
- explicit evaluation: offline -> guess query need/intent
  - bad: human error, inconsistent, inefficient, expensive; relevancy is continuous, subjective, situational, temporal
    - more assessors -> less relevant overlap
    - primary assessor: then no need other assessors
    - majority vote/intersection only
    - another assessor to resolve inconsistent evaluations
    - better: auto filter relevant -> human judgement (e.g. SE pooling), human judgement to train auto
- confusion matrix: retrieved (true) vs un-retrieved; relevant (true) vs irrelevant
- false positive, true negative, true positive, false negative
- recall = TP/(TP + FN): ability to find all relevant items
- precision = TP/(TP + FP): correlation of query to results, completeness of indexing algorithm
- relevant items usually unavailable
- precision—recall tradeoff
- problems with recall and precision: relevancy != system goodness, irrelevant items unconsidered, undefined when denominator is 0
- fallout = FP/(TN + FP): ~inverse of recall, denominator unlikely to be 0
  - least sensitive to searching accuracy, more reflecting of overall relevance to query
- good system: high recall, low fallout
- total number of relevant items: unknown for infinitely large collection (e.g. web)

    - estimate method 1: use a small sample, then extrapolate
    - estimate method 2: multiple algorithms on all documents, perform human judgement to filter on the union, then use the size of the filtered union (biased sampling)
- consider ranking: calculate the "moving" recall, (top-k) precision, and fallout (until recall or fallout is max)
    - precision (y)—recall (x) graph: looks decreasing zip-zag shape, start and end missing
        - interpolate: looks like decreasing staircase, start (x = 0) and end (y = 0) not missing
        - std 11 levels: 0.0, 0.1, ..., 1.0
        - multiple queries: take average -> smooth when many queries
        - average precision (AP): area under the interpolated graph;

$$\frac{1}{\#\text{ relevant}}\sum_{k=1}^{\#\text{ retrieved}} P@k \cdot \text{rel}(k)$$

        - mean average precision (MAP): average of AP for many queries
    - recall (y)—fallout (x) graph: similar to above, but increasing
- compare systems: curve closer to top is better
- single-valued measures: precision, recall, fallout
    - good: only reflect effectiveness, independent of cutoff, single number
    - bad: always consider all, ranking unconsidered (graph area partially solves this)
- F-measure:

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)\cdot P \cdot R}{\beta^2 P + R} \qquad \beta^2 = \frac{1-\alpha}{\alpha}$$

- F_1-measure: $\alpha = 0.5, \beta = 1$
- discounted cumulative gain (DCG): continuous relevance score;

$$\text{DCG}_p = \sum_{i=1}^{p} \frac{\text{rel}_i}{\log_2(i+1)}$$

    - normalized DCG (nDCG): ranges from 0 to 1, IDCG is the ideal DCG (sorted by decreasing relevance);

$$\mathrm{nDCG}_p = \frac{\mathrm{DCG}_p}{\mathrm{IDCG}_p}$$

- reciprocal rank (RR): 1/(rank of first relevant document)
  - mean reciprocal rank (MRR): average of RR over many queries; = MAP when always 1 relevant doc
- subjective relevance measures
  - novelty ratio: ratio of relevant retrieved results that are new; usefulness
  - coverage ratio: ratio of known relevant results that are retrieved; to find something seen before
  - sought recall: examined / (examined & relevant); precision and UI
- other factors: user effort, response time, UI, collection coverage

benchmarking

- analytical performance difficult, too many factors -> benchmarking, environment-sensitive
- benchmark collection: standard docs and queries, relevant docs for each query
  - SMART collection: ftp://ftp.cs.cornell.edu/pub/smart, Text Retrieval Conference (TREC): http://trec.nist.gov/
- problems: benchmark-specific, resource-consuming, web/Asian languages, expose weakness
- collections were small; table below
- TREC: Defense Advanced Research Projects Agency (DARPA), National Institute of Standards and Technology (NIST); annual since 1992; submit P/R values and present results in conference
  - objectives: compare IR techniques, benchmark development collaboration, encourage participation from industry and academia, development of new evaluation techniques (retrieval, routing, filtering, non-English, web-based)
  - good: large, judgements provided, supported by the US gov, wide and increasing participation (28 papers/360 pages@1992 to 112 papers@2006)
  - tasks
    - ad hoc: asking new questions
    - routing: asking same questions for new information
    - > TREC 5: interactive, multilingual, natural, database merging, large corpus (20 GB/7.5M docs)

    - relevance judgement: no exhaustive (100 topics * 742611 docs), no sampling (per topic, relevant avg 200, max 900), maybe polling (per topic, 2398 docs (33 runs of top 200), 1932 docs (22 runs of top 100))
    - evaluation: summary table (# topics, # docs retrieved & relevant), recall—precision avg (AP graph at std 11 levels), doc-level avg (AP when 5, 10, 15, 20, 30, 100, 200, 500, 1000 docs retrieved), AP histogram (in a topic, compare AP of a system against median precision of all systems)
- good systems: easy to use, reliable, effective, inexpensive

name	# docs	# queries	size (MB)
CACM	3,204	64	1.5
CISI	1,460	112	1.3
CRAN	1,400	225	1.6
MED	1,033	30	1.1
TIME	425	83	1.5

text processing

- IR: doc -> tokenize -> stem -> stopword -> index
- NLP: doc -> tokenize -> lemmatize -> part-of-speech parsing -> stopword -> NER, etc. -> index
- text as: string, words, linguistic units
    - single term works well: phrase are too specific
- index: tokenize -> stopword -> stem -> replace by term IDs -> count TF -> (optional) related terms for low TF -> (optional) phrases for high TF -> compute weights -> inverted & forward indices
    - query: tokenize -> stopword -> stem -> replace by term IDs -> compute weights -> query vector -> compute similarities -> ranking
- stopword, stem: search = index, search can be more restricted for stopword
- stem: unify variations, smaller index, enhance recall
    - improve IR, not lingustic; correctness is much less than 100%; incorrect stemming (e.g. fries -> fr)
    - why no stem?: general SE generally do not, domain-specific do
- expand query terms: no stem, query become many terms OR-ed
    - precise but more time consuming

- most: stem on both, suffix truncate more common than prefix, avoid over and under
- stemming methods: automatic stemming -> affix removal, successor variety, table lookup, N-gram; affix removal -> longest match, simple removal, context sensitive
- table lookup: term -> stem; space may be big; hard to capture all possibilities or context
- affix removal: remove suffixes (e.g. -ses, -ation, -ing)
  - longest matching once; multiple times; avoid: ability -> NULL, sing -> s
  - bad: linguistic knowledge, cannot cover all cases, language dependent
- context: consider other words
  - example: Ati -> Aty, where A is context, then only apply this transform if A contains vowel
  - many exceptions
- Porter's algorithm (1980)
  - 60 suffixes grouped into 5 steps
  - context-free and context-sensitive rules
  - FB93, Chap 8
- corpora-based statistical stemming: statistical analysis of a corpus
  - good: auto and language independent
  - assumption: stems rarely change, variations are based on stem
  - text mining: articles -> analysis/mining -> mapping rules
- successor variety: number of possible characters follow a string in a corpus (end-of-word is a character)
  - a prefix has low variety -> unlikely stem
  - cutoff method: segment when variety >= threshold (2)
  - peak and plateau method: character whose variety is greater than preceding and following character
  - entropy method: absolute variety does not consider frequency
    - calculate information entropy: b = 2 -> in bits;

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log_b(p(x))$$

      - English entropy: 1.0~1.5 bits/0.6~1.3 bits
    - $p(x)$ is simply the probability of a character
    - then combine with cutoff method or peak and plateau method

    - select segment as stem
        - heuristic method: 1st or 2nd; if 1st is likely prefix, choose 2nd
        - complete word method: complete word
- shared bigram stemming
    - 2-gram example: abcdef -> ab bc cd de ef
    - similarity of 2 terms: Dice coefficient

$$S = \frac{C}{A+B}$$

, where C is # shared unique bigrams, A, B are # unique bigrams for the 2 words individually

    - 3-gram/trigram
    - cluster terms using coefficient, terms in a cluster probably has the same stem
- efficiency: ~43%, precision becomes lower

relevance feedback

- why: user types few things, ambiguity
- query reformulation: adjust query weights or replace words
- query expansion: add terms
- manual feedback: by user; difficult because too much effort
- auto feedback: improve query using relevant assessments
    - explicit: explicitly mark as relevant and irrelevant; but users too lazy
    - implicit: user actions (clicks)
        - provide additional info: related pages, related queries, groups of docs
- accept user relevant judgement -> feedback formula
- reinforce relevant results, weaken non relevant results
- optimal query: let D'_R be the relevant docs centroid, D'_N be the irrelevant docs centroid;

$$Q_{\text{opt}} = C(D'_R - D'_N) \qquad C \text{ is an arbitrary constant}$$

    - all relevant and irrelevant docs are not known
    - estimate:

$$Q' = \alpha Q + \beta D'_R - \gamma D'_N$$

        - positive: relevant only; negative: irrelevant only; mixed: both

        - used docs: all, all relevant and highest-ranked irrelevant
        - doc vector: only use some, high weight terms, common terms in all relevant docs and common terms in all irrelevant docs
    - relevant more valuable than irrelevant -> $\beta > \gamma$
    - effective when relevant docs are clustered
- doc modification: move relevant closer to query and vice versa for irrelevant
    - relevant
        - query term not in doc added with factor α
        - query term in doc added with factor β
        - doc term not in query decreased with factor γ
    - irrelevant
        - doc term also in query decreased with factor δ
        - doc term absent from query increased with factor ε
    - persistent changes to docs
    - makes user input part of doc weights
    - makes ranking changes all the time
    - bad: docs not identified as relevant become less accessible by general queries, not repeatable results
    - what if docs come from separate clusters?
        - solution: split query vectors by user feedback
            - rarely used: need many relevant docs, doc space mod is expensive, how to mod docs?, difficult performance evaluation

personalization

- why: one search fits not all
- capture user interests, then re-rank
- explicit & implicit (see above)
    - implicit good: easily collectable, reflective of real usage
- clickthrough, browsed docs
- absolute feedback: results that are relevant or irrelevant
    - problem: click != relevant, click paradox: user trust SE -> SE trust user
- relative feedback: results that are preferred over other results
    - NOT clicked, more reliable than absolute, 80% are correct
- log clickthrough, and many other things, how to model?

- eye tracking, fixation: 200-300 msec, saccades: 40-50 msec, pupil dilation
  - results: golden triangle at top left; organoic: 100%@1 -> 20%@10, ads: 50%@1->10%@8
  - main content at top left
  - fixation correlated with clicks, first link > second link even fixation similar
- clickthrough analysis: record rank and pages that are clicked; better for relative feedback
  - assumption: read sequentially, click relevant, skip irrelevant
  - results: relevant > irrelevant; but unknown comparing among relevant themselves, need strategy
    - (1) click > skip above
      - penalize highly ranked unclicked pages
      - first not affected
      - no: clicked <=> clicked, unclicked <=> unclicked
    - (2) last click > skip above
    - (3) click > earlier click (order of clicks, not necessarily ranking)
    - (4) click > skip previous (backward 1 only)
    - (5) click > no-click next (forward 1 only)
  - strategy performance
    - 80%-90% correlation: (1), (2), (4)
    - 65%-75% correlation: (3), (5)
- apply preference to ranking
  - number of preference rules are sparse considering many webpages
  - turn preference rules into concept/keyword preference, then can handle unseen webpages
  - since has conflicting conference, use machine learning is best
  - implementation: middleware that stores user profiles, preference vector
    - effect: re-ranks the returned results, or query reformulation (maybe adding the preference vector)

Zipf's law

- indexing is also important: semantics, importance
- TF×IDF theoretical basis

- Zipf's law (1949): freq ∝ 1/rank
  - principal of least effort
  - application: affected inverted index size and lengths, find stop words, explain content words and polysemy
    - self-reinforcing phenomena (e.g. in-degree of nodes)
- Mandelbrot's formula (1954): log(freq) = log(P) - B * log(ranking + ρ)
- Zipf's other observations: short frequent words, most frequent 20% words: 60% usage, number of meaning ∝ sqrt(f)
  - content words: at most 24 times in corpus
  - $F \propto I^{-p}$; p between 1 and 1.3, F is the # two content words appear I interval apart
- power law: generalization of Zipf's law: $y = ax^k + o(x^k)$ <=> log y = k log x + log a, k is scaling component
  - scale invariant

index term selection

- historical: few, computer: all words: web: too many words
- index all terms
  - too much noise, may overwhelm content
- choose terms with high TF×IDF weights
  - arbitrary threshold, number of terms, different weighting schemes
- choose terms from predefined dictionary
- too ad hoc
- term POV: content: meaning, index: docs containing words, search: find relevant docs
- some term discriminate relevant docs better than others
  - in a vector space: spread out the docs
    - adding a new term: not contain -> unchanged, contain -> dragged further away
- decision tree can represent containing terms
- good index term decrease average similarity
  - average similarity: group average linkage;

$$Q = \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{j=1, i \neq j}^{N} \text{sim}(D_i, D_j)$$

    - bad: expensive, requires recomputation

    - term discrimination value (TDV): old average similarity - new average similarity
        - TF×IDF can be TF×TDV
        - bad: does not consider dissimilar docs
    - relationship with TF: usually positive, and increasing initially, then decrease and eventually become negative increasing slowly
        - low TF but initially increasing: make docs containing them extremely close together
    - average similarity (compromise): average distance to centroid;

$$Q = \frac{1}{N}\sum_{k=1}^{N}\text{sim}(C, D_k)$$

- greedy and exhaustive term selection algorithm: identify words -> select similarity -> set term weights to 1 -> all terms are candidates -> compute DV for all -> select terms with highest DV -> removed selected terms from candidate -> repeat until enough terms

co-occurrence

- simple text analysis assumption: human writings are not random
- N-grams
    - 2-grams example: alpha beta gamma -> alpha beta, beta gamma
    - mostly meaningless, many unique N-grams
    - good: faster retrieval, improve recall, hurt precision
    - N = 2 or 3 for English
- remove bad N-grams
    - if contains stopwords
    - if below threshold, need to set threshold, does not stop high-freq stopwords and phrases
    - grammatical filtering
        - prefer nouns, noun phrases, and compound nouns: N-N, A-N, (A/N)-*-N
            - adjective: A, noun: N, prepositions: P
        - bad: requires part-of-speech tagging
    - combine filters
- collocation (NLP) and co-occurrence (IR)
    - phrases are sensitive to word positions

    - count words appearing in the same sentence, paragraph, document, or n-word window
    - good: considers "near" rather than "consecutive", reveals usage patterns
    - application: find meaningful phrases, sentiment analysis, language study
    - compositional collocation: if can predict meaning
        - non-compositional collocation: likely NN pairs
- co-occurred terms: same doc
- co-occurred window: n-word window
    - distance between 2 words: directional (signed), words between them + 1, "Donaldson's" is 3 words
        - use: mean and variance of distances
- raw co-occurrence frequencies may be deceiving (base rate fallacy)
- pointwise mutual information (PMI): normalize co-occurrence frequency;

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

    - 0 when uncorrelated, +ve is correlated, -ve is anti-correlated
    - applications: reveal real word knowledge, query suggestion, identify page topic, page summarization
        - page ranking: promote pages having correlated terms
        - spam pages: many popular terms but little correlated terms

enterprise SE

- Google Mini: 50k docs for US$1995, up to 300k docs
- Google Search Appliance (GSA): 500k for US$30000, up to 30M/billions docs
    - hardware + software: Linux, Intel CPU, Dell
    - bad: little support, unimpressive search quality, limited regions, 1% of gross revenue
- enterprise SE forecast: US$717.2M in 2006, US$3.8B in 2020, US$8 in 2027
- history
    - past leaders: Microsoft, Endeca, Autonomy (acquired by HP in 2011-10 at US$10.3b)

    - 2011: Microsoft, Endeca, Google
    - Microsoft’s enterprise SE based on FAST
- insight search and insight engines: natural access to information
    - additionally describe, discover, organize, analyze, gather content from many sources
    - uses natural language and rich context, proactive > activated by user
    - deliver actionable insights from full spectrum of content and data
- enterprise search challenges: critical, demanding, mission-critical
    - challenges
        - priority: correctness > relevance (CEO expects first result to be correct)
        - much fewer but unmissable answers
        - queries are much more specific (e.g. car sales in this summer, feedback about a new car model)
        - links do not work because little users, little links, and for navigation
        - not web-based (e.g. DBMSs, folders on network drives, PDF, Word)
        - security: SSL, user login, access control, summary allowed for protected pages
        - flexible scoring to fit corporate structure and promotion campaigns
        - classify pages (e.g. product, press release)
        - search pages; also people, expertise, and other resources
    - exploitable
        - little data compared to the web
        - domain-specific, higher quality results and predictable queries
        - no spam
        - well-organized docs (e.g. similar docs in folders)
        - more analyzable user behavior (e.g. user IDs and page clicks)
    - example: recommendation/parametric search, selectable collections, experts, related docs & cats & queries
    - advanced requirements
        - taxonomy, classification, clustering

        - information extraction (e.g. named entities, concepts, syntactic analysis, statistical analysis)
        - web-based SE management interface (e.g. start/stop, configure ranking & cats)
        - context-sensitive search (e.g. based on employee type)
        - SE analytics: frequently asked queries, queries with no results, pages with no clicks, average rank of user clicks (ARUC)
        - community-based and usage-based ranking
            - users from the same department look for similar things: user or group sensitive ranking
            - challenge: identify groups independent of corporate structures
                - solution: personalization (keyword vector)
            - promote frequently clicked pages
    - applications: clustering, knowledge engine
- key components
    - more functions: federate, search, browse, alert, recommend
    - security, secure gateways
    - handling different data sources