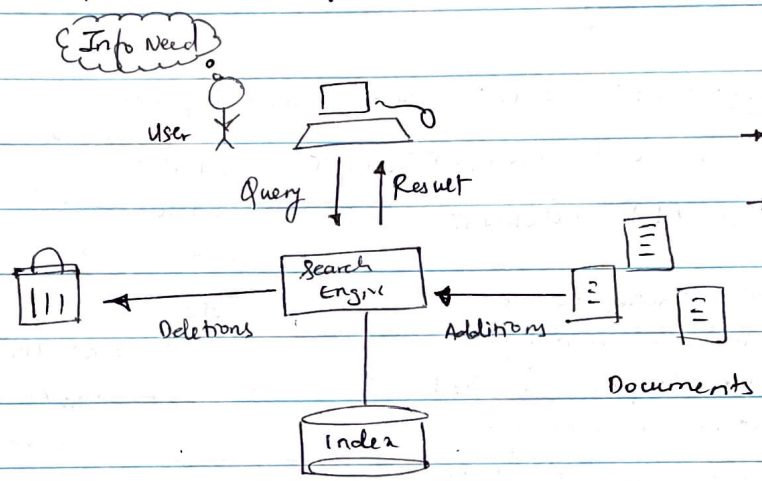
INFORMATION RETREIVAL

* Represent, search, manipulate large collection of data.

* Finding data from unstructured material.

* -Corpus: large repo. of docs. (majority ∈ documents)] Need expressed in term of query
  -Info Need: topic about which info req^d



→ Documents are indexed.
→ According to Relevance results are returned.

* Types of Data → Structured : Relational Database -
  - Tabular format - Highly organized
  - Rows & cols related - Easy to store, process, access

  Unstructured Data : textual, number, dates, BLOB (Binary large objects)
  - not organized in predefined manner.
  - 80% worldwide data
  - Require more storage.
  - Rich media (video, image, audio), analyzable.
    eg: Phone calls, log files, Social media comments, images.

  Semi Structured Data
  - not complete organized structure for relational database
  - self describing structure tags, attributes to seperate entities within data
    eg: XML.

* Structured: Technology, Transaction management, Version Management, ~~less~~ ↓ scalable
  ↑Flexibility, Query Performance ↑ (complex).

* Unstructed : ↑scalable, Not complex.

* to avoid linear scanning of large documents, we need to index the documents in relevance.

INVERTED INDEX

TDI: Term Document incidence matrix

TDI Matrix infeasible when record too large. (Usually Sparse Matrix)

Hash map type

Index data structure, stores mapping from content (words or docs) to its location in document or set of documents.

Steps for Building Inverted Index:

→ Token Sequencer: Sequence of each term in each doc, given doc id.

→ Sorting: alphabatically.

→ Group: by words → by doc id

→ Splitting into dict & Posting: store terms and pointer to posting list for each term.

Can be stored in dynamic array

Consume less mem

BOOLEAN QUERY RETRIEVAL: Query in Boolean Expression of terms.

Terms combined with AND, NOR, NOT

eg: BRUTUS AND CALPURNIA

Steps:

1. Locate Brutus in Dict
2. Retreive its posting
3. Locate Calpurnia in Dict
4. Retrieve its posting
5. Intersect both the lists:

ALGO:-

```
answer ← < >
while p1 ≠ NIL & p2 ≠ NIL
do
    if docID(p1) = docId(p2)
        ADD(answer, docId(p1))
        p1 ← next(p1)
        p2 ← next(p2)
    else if doc p1 < doc p2
        p1 ← next(p1)
    else p2 ← next(p2)
return Answer
```

RETREIVAL MODEL:

Process of matching a query and document.

Basis of Ranking in search engines.

Techniques: query suggestion, query expansion, relevance feedback, use interaction and context to refine

★ Important for retrieval models and evaluation measures. (key aspects of relevance)

- topical and user relevance.
- whether it is binary or multivalued.
  - binary: relevant / non relevant.
  - multivalued: relevant, non relevant, unsure

BOOLEAN RETREIVAL: exact match retreival

Assumption: All docs equivalent in terms of relevance.

Merit:- Results predictable, easy to explain
- Any feature of documents can be used as operand.
- Efficient in fast elimination

Demerit:- effectiveness on user.

∵ Equivalent relevance Results in sometime low or no retrieval of highly relevant DOC.

(may return less user relevant)

IR image 1 word doc.

VECTOR SPACE MODEL

query assumed as part of (t) dimensional vector space.
↑ number of index terms

Document D_i represented by vector of index terms.

$D_i = (d_{i1}, d_{i2}, d_{i3} \ldots d_{it})$

D_{ij} = weigh of j^{th} term

Query = vector of t weights.

$Q = (q_1, q_2 \ldots q_t)$

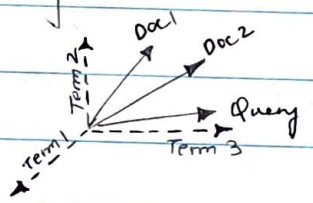
q_j = weight of j term in query.

	Term 1	Term 2	...	Term t
Doc 1	d_{11}	d_{12}	. . -	d_{1t}
Doc 2	d_{21}	d_{22}	. . . .	d_{2t}
:				:
Doc n	d_{n1}	d_{n2}	. . -	d_{nt}

Stop words not indexed

words have been stemmed

Range 2



Documents can be ranked by computing distance bet^n points representing the documents and query.

Similarity measure used to calculate distance.

Cosine: angle bet^n query and doc vectors.

vectors normalized, represented by vectors of equal length

Cosine of angle bet^n two identical = 1 angle = 0

cosine of angle bet^n two dissimilar vectors (non zero term) = 0, angle 90.

$$\text{cosine}(D_i, Q) = \frac{\sum_{j=1}^{t} d_{ij} \cdot q_j}{\sqrt{\sum_{j=1}^{t} d_{ij}^2 \cdot \sum_{j=1}^{t} q_j^2}}$$

* Tokenization:

Chopping text into tokens (pieces), throwing away certain characters

Token: useful semantic unit

Type: class of all tokens containing same char sequence.

Stemming → Term: Type that is included in IR system dictionary

* Issues with Tokenization:

- characters like , ' : are thrown away, this may result in Data loss ex: apostrophee
  O'Neill.
  Los Angeles

- Language issues: eg: French,
  German write compound nouns without spaces -
  (compound splitter module should be used (subdivide into multiple words, for words present in vocabulary).

↳ East Asian languages: no space between words.
  word segmentation can be used as prior linguistic processing

* Stemming: Reducing inflected words to word stem, or root form.
  Applications: IR, determine domain vocabulary in domain analysis.
  Algo: Potters stemmer (5 rule algo)
  Dawson "
  Lovins "
  Xerox "
  Krovetz "

* Stop words: common words, ignored when searching and retrieving result.
  using stop list reduces number of postings. (POSITIVE)

* Term Frequency - Inverse Document Frequency (statistical measure):
  evaluates how relevant a word is to a document in collection of docs.

metrics:-

tf ⇒ ~~term frequency~~ of word in doc.
idf ⇒ of word across a set of documents. Closer it is to 0, more common a word is.

if word not common in docs, to value comes near to 1

tf * idf = score (weightage of particular doc)

↑score ↑relevant word in particular document

$tf\ idf(t, d, D) = tf(t, d) \cdot idf(t, D)$

(t: term, d: particular doc, D: set of docs)

$tf(t, d) = \log(1 + freq(t, d))$

$idf(t, D) = \log\left(\frac{N}{count(d \in D : t \in d)}\right)$

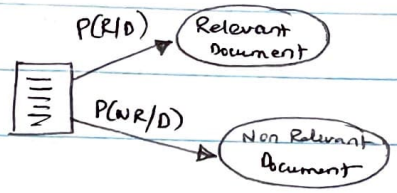
N: total documents.

* ~~Language Model~~: Probabilistic Model :

when info is returned to user, highly ranked docs will be displayed first, i.e they have highest probability. This probability goes on decreasing as we move down the response

How to calculate probability of relevance?

Treats info as classification problem.



Model classifies, then returns if relevant.

P(R/D) - conditional prob (Prob of relevance, given representation of document)
Similarly P(NR/D).

P(R/D) → P(D/R) } Proba of doc being in relevant set.

Bayes Rule.

$$P(R/D) = \frac{P(D/R) \cdot P(R)}{P(D)}$$ ← normalizing constant

Relevant if $P(D/R)P(R) > P(D/NR)P(NR)$.

$$\frac{P(D/R)}{P(D/NR)} > \frac{P(NR)}{P(R)}$$

Score { likelihood ratio → $\frac{P(D/R)}{P(D/NR)}$; $\frac{P(NR)}{P(R)}$ }

relevant set

* Language Model :

Topical Relevance } represent text in various language technologies; speech recognition, handwriting recognition...

Simplest language model unigram language model.

If we have prob of each term in a document, we can predict what the next word in sequence would be.

Natural log

n gram predict words on basis of previous (n-1) words.

words imp for topic will have unusually high probability.

* Query ~~Likelihood~~ Likelihood Ranking :- probability of query generation is the measure of how likely is that document is about same topic as the query.

Issue if word missing as it will give probability 0.

$$P(D/Q) \overset{\text{rank}}{=} P(Q/D) \cdot P(D)$$ ← uniform

Ignore P(Q) as denominator normalizing const

$$P(Q/D) = \prod_{i=1}^{n} P(q_i/D)$$

$$P(q_i/D) = \frac{f_{q_i,D}}{|D|}$$ ← frequency of query words in D
← number of words in D

Refer language modelling ppt

* Okapi : The BM25 Ranking Algorithm

BM: Best Match

Ranking Algo / scoring value : How query related to document.

Relevance of doc to search query.

BM25 used in probabilistic framework as core part

Simplest score for document d = idf weighting of query terms present in d.

Refer Image under 4

$K = k_1((1-b)+b(ld/lave))$

↑ score ↑ relevance.

$$RSV_d = \log\left(\frac{N}{df}\right) \cdot \frac{(k_1+1)tf_{td}}{k_1((1-b)+b(ld/lave))+tf_{td}}$$

ln natural log

$$= \log \frac{(r_i+0.5)/(R-r_i+0.5)}{(n_i-r_i+0.5)/(N-n_i-R+r_i+0.5)} \cdot \frac{(k_1+1)f_i}{K+f_i} \cdot \frac{(k_2+1)qf_i}{k_2+qf_i}$$

* RELEVANCE FEEDBACK

Polysemy, synonyms ← same word diff meaning → Helps overcoming the problem

eg: plane fuel, fuel for aircraft, jet, ?

- Global methods for expanding or reformulating query.
  - Thesaurus generation ~~for~~ automatic query expansion
  - Spelling correction.
  - Query expansion with Thesaurus.
- Local method : adjust query relative to the doc that initially appear
  - Relevance feedback
  - Blind/Pseudo Rel feedback
  - (Global) Indirect Relevant feedback

Assume: Indexed content unknown to user. Info need changes after looking at result

Relevance Feedback

- user query
- Initial set returned
- user selects relevant
- System recomputes better results based on users feedback
- System display revised results

Relevance feedback : Rocchio Alg

Find query vector q, maximise similarity with relevant docs.

Rocchio: $\vec{q}_{opt} = \arg\max [sim(\vec{q}, C_r) - sim(\vec{q}, C_{nr})]$

Similarity bet^n q and set of relevant docs.

↑ recall

+ve & -ve feedback

values $\gamma < \beta$ using cosine similarity.

$$\vec{q}_{opt} = \frac{1}{|C_r|} \sum_{\vec{d}_j \in C_r} \vec{d}_j - \frac{1}{|C_{nr}|} \sum_{\vec{d}_j \in C_{nr}} \vec{d}_j$$

diff bet^n centroid of relevant and non relevant doc vectors

Image (6)

Negative terms are taken 0.

In practice

$$\vec{q}_m = \alpha \vec{q}_0 + \beta \frac{1}{|C_r|} \sum_{\vec{d}_j \in C_r} \vec{d}_j - \gamma \frac{1}{|C_{nr}|} \sum_{\vec{d}_j \in C_{nr}} \vec{d}_j$$

α, β, γ weights attached

RF is expensive,

long queries expensive to process.

often hard to understand why some particular doc retrieved after RF.

* Evaluation:

Required before deploying

we need test collection

1. Document collection
2. Info need, expressible as query
3. A set of relevance judgements, standardly a binary assessment

Gold Std / Ground Truth Judgement of Relevance.

No parameter Tuning.

Relevance is assessed relative to information need and not query.

Final decision on user

* why evaluate?
  - effectiveness & efficiency: to find right information in the quickest time.
  - cost (processor, memory, disk space, networking).
  - comparision bet^n techniques.

Traditional Effectiveness Measures :-

Assumption

- a doc in text collection would be relevant or irrelevant
- Relevance depends on info need and d, and independant of other docs.

$$\text{Precesion} = \frac{\#(\text{relevant items retreived})}{\#(\text{reteived items})} = P(\text{relevant}/\text{retreived}).$$

$$\text{Recall} = \frac{\#(\text{relevant items retreived})}{\#(\text{relevant items})} = P(\text{retreived}/\text{relevant})$$

↑ Search Engine

$$P = \frac{tp}{(tp+fp)}$$

↑ Analysis document

$$R = \frac{tp}{(tp+fn)}$$

	Relev	Non Relev
Retrein	tp	fp
Not Retr	fn	tn

$$\text{Accuracy} = \frac{tp+tn}{tp+fp+tn+fn}$$

In almost all circumstances, data is extremely skewed., 99.9% non relevant

* F-Measure : weight harmonic mean of P & R.

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R} \quad \text{where } \beta^2 = \frac{1-\alpha}{\alpha}$$

$\alpha \in [0,1]$ $\beta^2 \in [0, \infty]$

default balanced F measure equally

$\alpha = \frac{1}{2}$ $\beta = 1$

$$F_{\beta=1} = \frac{2PR}{P+R}$$

* Latent Semantic Indexing :-

* Summarizing via index terms can lead to poor retrieval due to two effects :
  - many unrelated docs might be included in result
  - relevant documents not indexed by any query or keyword are not retrieved.

due to inherent vagueness with retrieval via keywords.

* LSI is an attempt to address the above issue.
  It tries to match the documents based on the concept matching instead of index term matching term.

In this retrieval Matrix computation is used as the basis for information retrieval. Matrix computation in particular Single Value Decomposition (SVD) is used.
This filters out noise found in documents such that two documents have the same semantics.
In a multidimensional space these docs will be located close to each other inspite of not having common terms.

Term doc matrix A, loc(i, j) (row, column) represent, no. of times term i appears in doc j.

SVD of A results in matrices $U \Sigma V^T$ — Σ is diagonal matrix

values in Σ = singular values. } sorted by magnitude, top k values taken as means of developing 'latent semantic' representation of matrix A.

Remaining singular values set to 0.

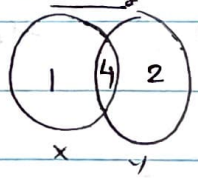
first k cols are kept in V_k

first k rows kept in V_k^T

After setting results to 0.

A' generated to approximate $A = U \Sigma V^T$

working for JC.



* Ranked Retrieval :-

- system returns an ordering over the (top) documents in the collection for a query.
- Free text queries : just one or more words in human lang. No operators or expressions.

Assign a score say in [0, 1] to each document. } shows how well document & query match.

Assign score to query document pair -
If query term not in document, score should be 0.
More freq the term higher the score -

* JACCARD COEFFICIENT :

jaccard (A, B) = $|A \cap B| / |A \cup B|$
jaccard (A, A) = 1 jaccard (A, B) = 0, if $A \cap B = 0$.

★ Issues with Jaccard

- privileges shorter docs.

  More sophisticated normalizing of length req^d $|A \cap B| / \sqrt{|A \cup B|}$

- does not consider ~~tf~~ term frequency.

- does not account for term informativeness.

  ↑ Imp of term in doc.

★ Log frequency weighting:-

★ weight of term t in d is

$$w_{t,d} = \begin{cases} 1 + \log_{10} tf_{t,d} & \text{if } tf_{t,d} > 0 \\ 0 & \text{, otherwise.} \end{cases}$$

Score for doc query pair; sum over terms t in both q and d:

$$\text{score} = \sum_{t \in q \cap d} (1 + \log tf_{td})$$

★ idf weight

df_t : no. of docs containing t.

↳ inverse measure of informativeness of t.

$df_t \leq N$

$$idf_t = \log_{10}(N/df_t)$$ } log dampens the effect of idf

Best known ↑ with ranking of term ↑ no. of occurrency in doc

tf-idf weight is the product of its tf weight and its idf weight.

$$w_{t,d} = \log(1 + tf_{t,d}) \times \log_{10}(N/df_t)$$

$1 + \log tf_{t,d}$

↳ affects queries containing ≥ 2 terms.

Score for doc given a query -

$$\text{Score}(q,d) = \sum_{t \in q \cap d} tf \cdot idf_{t,d}$$

* Language Modelling (Cont.) *

Query likelihood: major problem, if any words missing from doc, score given will be zero.

Smoothing used to avoid this:

lower probabilities of word seen in doc, and assign the rest of words not seen

Estimate for unseen words based on freq occurrence of words in doc collection.

$P(q_i/c)$ = P for query word i in collection of docs c.

estimate = $\alpha_D P(q_i/c)$

α_D = coeffs of controlling prob assigned to unseen words

Prob estimate for word seen is $(1 - \alpha_D) P(q_i/D)$

Dirichlet smoothing ← depends on doc length } more effective

$$\alpha_D = \frac{|D| + \mu}{\mu}$$

μ set empirically { best results when $1000 < \mu < 2000$ }

$$P(Q|D) = \sum_{i=1}^{n} \log \frac{f_{q_i,D} + \mu \frac{c_{q_i}}{|c|}}{|D| + \mu}$$

c_{q_i} = no. of times query word occurs in collection of docs.

$|c|$ total no. of word occurrences in collect

$f_{q_i D}$, freq of q_i in D.

$|D|$ no. of words in D.

* Errors in stemming

over stemming: university & universe

under stemming: data & datum

↑

could be reduced to datu ← wrong

* Access Time less for memory
* Storage high for Disk

Image 5