INFORMATION RETRIEVAL (pt.1)

Day / Date: _____

CHAPTER 1 – BOOLEAN RETRIEVAL

Information Retrieval – finding material (usually documents) of an unstructured (usually text) that satisfies an information need from within large collections.

→ **Unstructured Data** – Data which doesn't have clear, semantically overt, easy-for-comp structure.

*IR is also used to facilitate 'semi-structed' ~~data~~ search (to search a doc where title was java & body was threading)

IR vs DB → Differences

① Format of data – DB: structured & clear semantic
IR: semi structure

② Queries – DB: SQL (formal)
IR: expressed in natural lang.

③ Result – DB: Exact result
IR: sometimes relevant

Text vs. Data

	Data Mining	Text Mining (Goal – exploit info; process of exploration data analysis that leads to new info.)
Data obj →	Numerical struc. + categorical data	Textual data (unstruc + semistruc) (but have a hidden structure)
Data represent →	Straightforward	Complex
Goal →	to extract info from a dataset & transform it into understandable struc. for further use.	– Item set mining } types of text mining/ method. – Classification – Clustering – Recommendation

Clustering → coming up with good grouping based on content of documents.

Classification → which classes each of the set of doc. belongs to.

Dazzle

Day / Date: _____

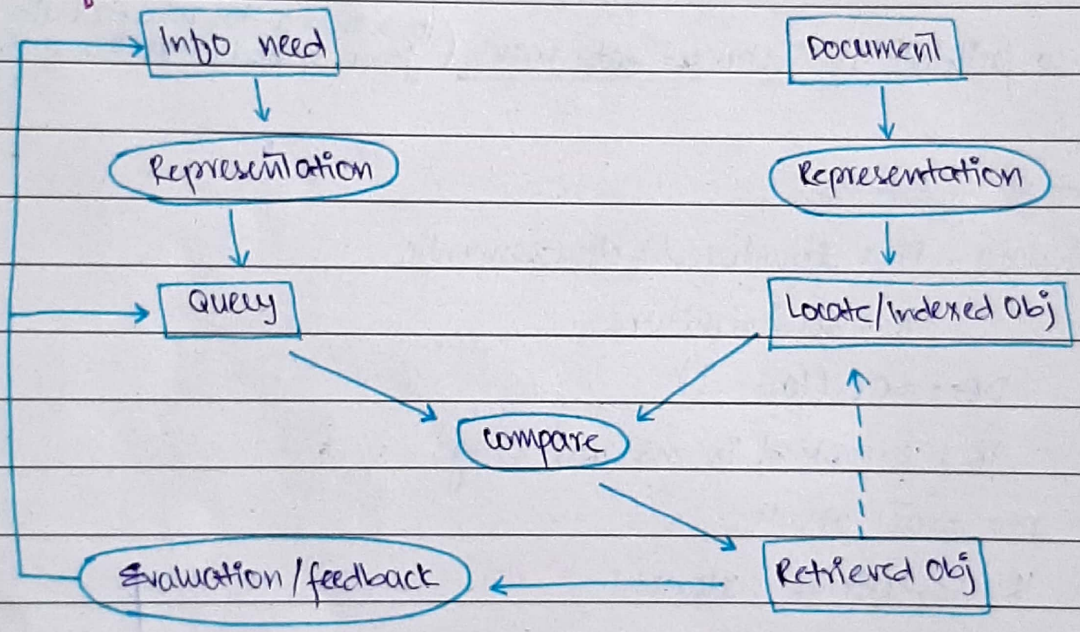
IR system can also be distinguished by the scale at which they operate.

① web search ② space of enterprise, institutional & domain specific search

③ personal info retrieval

Information retrieval vs. Text Mining

Basic IR process



Representation of both can be diff. so we bring them on same level by extracting something common & then compare them

→ Indexing is done before the query is applied so that all (eg. frames) can be indexed & Its easy to compare.

→ work on shallow features of documents — IR.

→ Retrieves / works on relevant text/document — text mining.

Content wise IR system

→ size (desktop, web scale)

→ Type (Text, multimedia)

→ Static (offline) or Dynamic

→ Exact match vs. Best match.

Info need → topic about which user desires to know more
Query → what the user conveys to the comp in an attempt to communicate the info need.

Day / Date: _____

Adhoc Retrieval Sys – Sys that aims to provide doc. from within the collection that are relevant to an arbitrary user info need, communicated to sys by means of user-initiated query. eg. Google, Bing.

Dimensions of IR	Applications for	Task
- vertical search	Images	Filtering
- forum search	Audio	clustering
- P2P search	Music	Content share / collaborative authoring
- literature search	Bibliographic search	

→ Image algo are diff. from usual search

① → QBIC – Query by Image content
↳ uses colour, texture, shape features for search & text based search.

② → Virage → supports queries based on color, layout & texture
↳ gives users more control over search process.

③ → VisualSEEK – research prototype.
↳ considers spatial relationship b/w obj
↳ matching spatial relationship b/w obj & visual features together gives powerful result.

→ Music search engine
↳ search by metadata eg. artists biography, music reviews.
↳ can create playlist, recommend, search for lyrics.

In Boolean model, we can simple scan-through doc & grep (linear scan). But we need more for other purposes
① To process large data
② To allow more flexible matching operations
③ To allow ranked retrieval.

Day / Date:

→ Mathematical Search Engine — to search mathematical content

→ Qs/Ans System — sys that allows extraction of ans to a request.

→ General Search Engine — Eg. google, yahoo, bing.

WHAT IS A MODEL?

- Abstract representation of a process or obj
  - → used to study properties, draw conclusion & make prediction
  - → Quality of conclusion depends upon how closely the model represents reality.

- A retrieval model describes human & computational processes involved in ad-hoc retrieval.
  - defines relevance, explicit
  - resource (info needed)

* BOOLEAN MODEL → based on set algebra. { set → collection of well defined & distinct obj
  - ↳ mathematical
  - ↳ features (dependency check & how we are modelling) } 2 dimensions
  - → pose any query which is in the form of a Boolean expression of terms which are combined with AND, OR & NOT.)

the user bring some word query & we have a resource D. { |D| = m (no. of doc.)

(w_i → string)

$D = \{d_1, d_2 \dots d_n\}$

d_i has $(w_1, w_2 \dots w_n)$

d_j is another document.

→ w_3 mein jitnay 1 hain wo doc aik jagah aagaye

$SR_B(q, D) =$ { }

search result ↓ Boolean ↓ query ↓ document we are retrieving

term ↓	d_1	d_2	...	d_n	← document
w_1	1	0		1	
w_2	0	1		0	
w_3	1	1		1	→ pick each row & aik jagah doc daal lein in sets mein
⋮	⋮	⋮		⋮	
w_n					

term doc. matrix ← (term x doc) → dimensions

Dazzle

Day / Date: _____

$SR_B = V_i$

→ Algebra (set, operation) → op should be well defined.
(int, decimal) → (×, ÷, +)

set of set alg. ← subsets
multiset → (has repetition)
↓ A = [a, a, b, b]

Boolean model is based on multiset aka (bag of features/word)
↳ if text → if img/sound

* linear or vector algebra, probabilistic algebra } from table in slides

→ Eminent way : Important
↓ There is a hidden relationship that we are not explicitly represent but we have captured it.
→ If represented explicitly → so we use models.

2 assumptions for Boolean model
① User can view doc & see features.
② we electronically view doc & see features.
↳ (form a dictionary)

{ 50% documents present → sparcity.
80% not present → not a good doc.

* 10^6 → max vector we can take for processing
→ Term Doc matrix mein rows le ke end kartay & ya koi aur operation.

Problems of Term Doc Matrix
→ Sparse matrix (most of the arrays are zero)
→ Exact matching
→ Complex Query Formulation.

Solution
→ Inverted Index

Day / Date: _____

Adhoc Retrieval System

Adhoc → user can bring any type of search.

→ Simple Interface when a text box to place query & search button.

→ Sys that aims to provide doc from within the collection that are relevant to arbitary user info need

Effectiveness of IR System

↓ quality of its search result

② factors of IR sys

1. Precision : what frac of returned results are relevant to info need?
2. Recall : what frac of the relevant doc in the collection were returned by system?

relevant / total

20 / 28 = 20/28

20 / 40 = 20/40

irrelevant / total

Precision	= relevant - retrieved / total retrieved
Recall	= relevant retrieved / total ~~retrieved~~ relevant collection

(learn)

100% recall → If we send all doc back 20/20 → retrieved / total = 100%.

100% precision → If we send 1 relevant result 1/1 = 100%.

8 → relevant

10 → irrelevant

20 → total relevant

→ precision = 8/18 = 0.44

→ recall = 8/20 = 0.4

Dazzle

Day / Date: _____

INVERTED ~~MATRIX~~ INDEX

① Dictionary / Feature space / Vocab

② Posting Listing

(Doc read krke uski terms nikaltay rahein ge)

list of doc present →

w_1 → Doc1, Doc3 → Doc6

(doc ids)

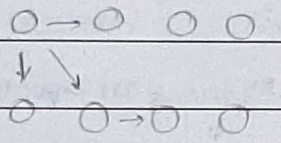
w_n

↓

linked list of terms

dictionary | posting list.

Intersect (Posting List Algo) → (works like merge func)



(we'll compare jab bhi aik list khatam na hojaye).

Query Optimization

↳ process of selecting how to organize work of answering a query so that the least total amount of work needs to be done by the system.

→ major element for this Boolean query is order in which posting list are accessed.

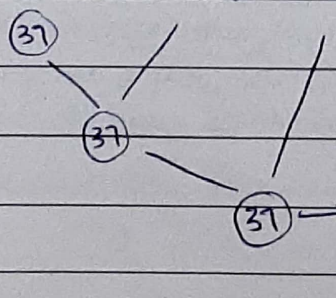
Intersect (Optimized)

(Sort bhi kr rahay hain. Jo chotti posting list hai wo pehlay ati hai)

we arrange terms so that we perform least amount of work. The no. of work depends on the shortest freq. of posting list.

t_1	t_2	t_3	t_4
1200	115	37	400

$(t_1 \wedge t_2) \wedge t_3) \wedge t_4)$



→ shortest freq among posting list.

```
Intersect (<t1...tn>)
terms ← SortByIncreasingFreq(<t1...tn>)
result ← postings (first(terms))
terms ← rest (terms)
while terms ≠ NIL and result ≠ NIL
do result ← Intersect(results, postings(first(terms)))
   terms ← rest (terms)
return result.
```

Day/ Date:

BOOLEAN MODEL

BM is most simple & most used. Considers that doc contains features (words, phrases) & the user's query is about these features.

Advantages

① Clean formalism, easy to implement & intuitive concept.

② Results are predictable, relatively easy to explain

③ Many diff. features can be incoporated.

Disadvantages

- Exact matching
- Query formation is hard
- All terms are equally weighted
  ↓
  (har feature equal weight ka hai)
- Flat result → all docs are at same level; all are equally important

→ Drawbacks of Boolean Model

① → no criteria for partial matching

② → No ranking of doc (grading scale missing)

③ → info need has to be translated into Boolean expression.

In conclusion → BM frequently results/return either too few or too many doc in response to a query.

④ Imposes binary criterion.

→ Extensions of Boolean Model

~~[illegible] / [illegible] / satisfied / ugly.~~

① Extended Boolean Model → to overcome the drawbacks of BM by making use of partial matching & term weights as in the vector space model. combines characteristics of vector space model with Boolean Algebra & ranks similarity b/w query & doc & so if doc maybe somewhat relevant to the queried term.

② Fuzzy Set Model
↓
Techniques are based on EBM; to define flexible sys ie. sys that can represent & manage vagueness & subjectivity which characterizes the process of IR & info representation.

Day / Date: _____

THE TERM VOCABULARY & POSTING LISTS (Chp 2)

* **Word** – a delimited string of characters as it appears in the text.

Term – A 'normalized' word (case, morphology, spelling etc); an equivalence class of words.

Error → in IR, collection of emails (dataset) → each file has mails of a single user ← (example)

* we need to find index of what we want to search.

Challenges of Document Processing

→ what is the format? (pdf, word) → what are hidden fields.
→ what is the language? (can be multilingual)
→ what characterset is in use? (single or multibyte?)
→ Size of document? (A file, email, group of files?)

Classification problems

* we need to convert the byte seq (file) into a linear sequence of characters.

Tokenization → Given a character sequence & a defined doc unit, tokenization is the task of chopping it up into pieces called tokens, perhaps at same time throwing away certain characters such as punctuations.

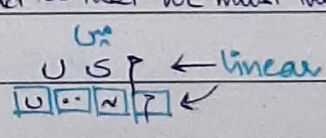
Token → an instance of a seq. of characters in doc that are grouped together as a useful semantic for processing.

Issues in Tokenization

(9/9/2019)

→ **Encoding**: (co-ed → 1 token ①; San Fransico → 2 token) or (learn format of numbers ②)
③ languages → (urdu, french) || we need to determine the correct encoding..

(we need to decide if a character makes a word so that we must throw a token)

The documents are usually linearized → 

Stop words → (to, am, the) → in isolation they don't give any semantic & are very frequent.
• with list, one excludes commonest word from the dictionary.
↓ but we need them for phrase queries (King of Denmark)
& relational queries (flight to london)

→ Indexing a large doc causes issue of index granularity.
↳ (can be handled by implicit/explicit proximity search)

Dazzle

Type → class of all tokens containing the same character sequence.

Term → type that is included in the IR system's dictionary.

Day / Date: → Issues of tokenization are lang-specific — lang identification based on classifiers that use short character subsequences as features is highly effective; most lang. have distinctive signature patterns.

Normalization (Equivalence classing of terms)

→ Each word should be changed to lower case (a general practice)

→ An abbreviation is mapped to a single word (hypens are removed)

→ The common or same words are tokenized to same token.

* Process of canonicalizing tokens so that matches occur despite superficial differences in char. seq. to the tokens.

→ MOST IMPORTANT CRITERIA

→ See how does your user write queries for words.

→ cedilla / diacritic (search)

Morphological Analysis

↳ Inflection → word change nai ho raha original / but kabhi change hojata hai. (who, whom)

↓ adding a suffix to a word, that doesn't change its grammatical category.

* such as tenses in verbs (-ing, -ed, -s); plural noun(s).

↓ don't change type of word.

(doesn't change type of word)

→ British & American English (colour, color) // use of thesaurus

→ Multiple languages

→ Dealing with complex words; foreign names → have unclear spelling or transliteration standard giving diff. spelling.

↳ we use heuristic to equivalence class or phonetic equivalents → best example is soundex Algo.

→ an alternative is

→ Thesaurus & Soundex

→ STEMMING & LEMMATIZATION

Reduce inflectional forms & derivationally related forms of a word to a common base form.

stemming → chops off the end of words or removal of derivational affixes to reach goal

lemmatization → remove inflectional endings only to return the base / dictionary form of word → lemma.

eg. saw → stem : s

lem : saw or see

Dazzle

(*tokens need to be indexed)

Day / Date: _____

walking, walks, walked → root is 'walk' (the base)

→ Each algo stem words on different basis.

→ Stemmer → general + fast → but the output doesn't make sense to user [drawback]

Porter's Algo is most common & effective algo for stemming.

Search engine → Stemmer; summarization of doc → lemmatization } depends on application.

* Stemmer → use lang. specific rules, but require less knowledge.

* Lemmatizer → complete vocab & morphology analysis to correctly lemmatize a word.

→ Stemming increases recall & while harming precision

POSITIONAL POSTING & PHRASE QUERIES

→ Phrase Queries

Want to be able to answer such queries as 'stanford university' → as a phrase.

$Q = (t_1, t_2)$ → both should be present & adjacent as well. t_1 should come first.

↓

Post Query Process (we'll check that t_1 & t_2 are adjacent in every doc)

↳ After result is generated, we need to validate our result. This increases work

Amount of work ∝ no. of doc in result.

Solutions

① Bi-word Index — consider every pair of consecutive terms in a doc as a phrase.

eg. friend roman countrymen } 2 bi words

① ②

↳ each bi-word is a dictionary term

↳ results are immediate.

→ (a long query requires a lot of work)

↳ so we'll divide it & then check from all documents (in dividing & then checking from all documents)

↓ (as a feature index)

have bi-gram index keeping for easy & then search it in posting list & make a posting list.

then scan in doc & make a posting list

Dazzle

Day / Date: _____

③ Make any string of words in form NX*N to be an E. Biword
④ Each such extended biword is made a term in vocab.

① Tokenize text & perform part-of-speech tagging.
② Group terms in nouns & other classes.

④ Extended Biword Index —

• used for searching some concepts: T X^n T → T search kartay hain par beech mein se kafi words X^n skip kardein gy.

eg. coins are in pockets
N X X N
↳ take out the nouns (N) & ignore preposition etc (X).
↳ lookup in index for 'coin pocket'

Issues in Bi-word

→ too many false positives → for which we'll have to do post indexing to validate that result is correct
→ no. of indexes increases → memory size increase (carefully place biwords in index)

↓
not a standard solution, but a compound strategy.

② Positional Indexes — most commonly used

* need to access the inverted index entries for each distinct term; has more info.
→ docID, freq & position of the term

* For each term in vocab, we store postings of the form docID: &position 1, pos 2.. & as token appears.
<term, no. of docs containing term>
↓ able to answer any length of phrase query

→ check that ① both terms are present in doc, as well as the positions of appearance in doc are compatible with phrase query being evaluating; use the following:

→ Proximity Query — (w1, w2, k) → & w1 & w2 dhaliye k words apart & nouns & gives the distance it needs a part.
(w1, X1, X2 .. Xk) → distance = k
phrase

→ positional that has a distance of K; all those docs will be in result.

eg. employment /3 : /k means 'within k words of (on either side)'

Positional Index Size → It expands required postings storage significantly.

Dazzle

(Biword se search (kuch common cheez ko) bohot saray result ayein ge aur unko merge krna is not preferred)

Day / Date: _____

phrase → phrase → to be or not to be

to	} posting list of each position	1:
be		2:
or		3:
not		44:

search for valid results in each doc.

positions at which each word is at.

Rules of Thumb

→ positional index is good for exp like queries

→ Positional index is 2–4 as large as a non-positional index.

→ Positional index size 35–50% of vol. of original text.

Combination Scheme

A combination of biword + positional phrase would do effective search.

→ Need to decide which method should be used for which query

* Where individual words are common but desired phrase is comparatively rare → most expensive query

> If posting lists are really long → we'll use skipping pointers at indexing time.

We walk through two posting list simultaneously; if long → time taken would be O(m+n)

Use skiplist by augmenting posting list with skip pointers (at indexing time).

↓ Effective shortcuts that allows us to avoid processing part of the postings list that will not figure in the search result. ↑ pointer comparision but longer span of skip.

① → where to place skip pointers ② how to do efficient merging using skip pointers.

* Presence of skip pointers only helps for AND queries & not for OR queries.

Where do we place skip pointers?

Tradeoff: • More skips means shorter skip span & that we are more likely to skip. But also mean lots of comparisions to skip pointers & lots of space storing skip pointers.

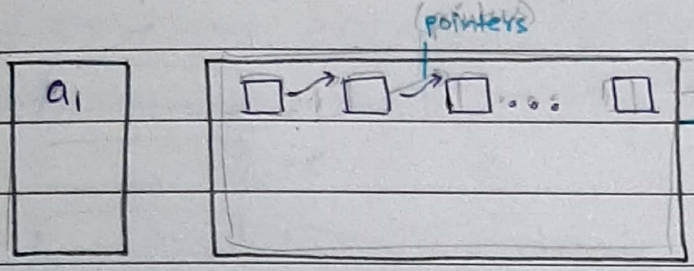
• Fewer skips means fewer pointer comparisions, but then long skip spans which means that there will be fewer opportunities to skip.

Heuristic → $P = \sqrt{\text{length of posting list}}$; use $\sqrt{P}$ evenly spaced skip pointers.

Dazzle

Day / Date: _____ Chp no. 3

DICTIONARIES & TOLERANT RETRIEVALS (Chp 3)



*dictionary is kept in memory for faster access

Dictionary

↳ determine whether each query term exists in the vocab, if so identify the pointer to the corresponding posting.

The vocab lookup uses the data struc. "dictionary" → two sol: ① Hashing ② Search Trees

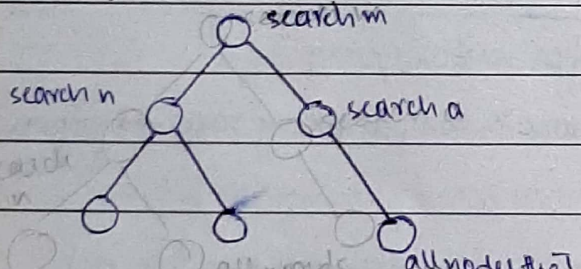
↓

The entries in the vocab (terms) are referred to as **keys**.

Data Structure for Dictionary

① Trees ② Hashing

↳



Trees overcome issues by:

→ allowing us to deal prefix

*Best tree is Binary Tree ~~or B Tree~~

↓ each internal node has 2 children

↓ searches with O(log M)

drawback → rebalancing

↓ as terms are inserted, or deleted from binary search tree, it needs to be rebalanced.

Hashtable

Each vocab term is hashed in as integer

Pros → lookup is faster than as for trees : O(1)

Cons → ① No easy way to find minor variant

② No prefix search (words starting with 'pre-')

x — x x — x

To reduce rebalancing, use **B-Trees**.

↓ a search tree in which every internal node has a no. of children in interval [a, b] where a & b are +ve integers; B-Tree can be viewed as 'collapsing' multiple levels of binary tree in one — advantageous when dictionary is disk resident // serves as prefetching binary test.

→ search tree demands that char. used in doc have a prescribed ordering.

Dazzle

Day / Date: _____

Hello $ → ending boundary

ello $ H → start from H & ello is suffix so hello is a word

suffix: lo $ Hel

prefix → starting from this

lo $ Hel → ending at this so lo is prefix for Hel.

→ WILDCARD QUERIES

① Trailing wildcard query (mon*) mon*

② Leading (*on) *on

③ General (m*n*tion) n*tion

→ efficiently perform these queries ; WHY DO WE NEED WILD QUERIES?

→ ① User is uncertain of the spelling of a query term (unintentional)

Sydney vs Sidney → S*dney

→ ② User is aware of multiple variants of spelling term & (intentionally) seek doc containing any of the variants (eg. colour vs. color)

→ ③ User seek doc containing variants of a term that would be caught by stemming, but unsure whether the search engine performs stemming (eg. judicial vs judiciary → judicia*)

→ ④ User is uncertain of the correct rendition of a foreign word or phrase (eg. Universit* Stuttgart)

mon* → use a simple tree following symbol m, o & n ; lookup in inverted index to retrive all doc starting with mon.

*mon → add terms to reverse trie create a reverse B-Tree & lookup then.

se*mon → se* → do by normal tree ; *mon → do by reversing B-Tree.

mon* → apply a ∩ b → we'll get desired result

General Wildcard Query

Two techniques to handle them:

① Permuterm Index

→ $: to mark the end of a term ; hello is written as hello $

→ create a permuterm index having various rotations of each term with $ that links to original vocab.

set of rotated terms → permuterm vocab.

* $ hagake rotate one letter at a time

* use B-tree as data structure. → cat $ → at $c, t $ca, $cat

that allows us to select rotation with a given prefix

* dictionary → dictionary size becomes large (as it has all rotations)

→ we'll keep permuterm index in the memory along with dictionary so we don't have to access it from disk

rotations need to map on one word

↓ create an index above our dictionary

↓ map rotations on a word (prefix/suffix) on our dictionary.

Dazzle

Day / Date: _____

② Bigram: k-gram indexes for Wildcard Queries

→ Chop entire term in k-characters at a time (k=2 if bi) {sequence of k-characters}

→ In a k-gram index, the dict. contains all k-grams that occur in any term in vocab. Each posting list points from a k-gram to all vocab terms containing that k-gram.

Disadvantage: Needs post filtering (post validation of result doc) b/c gives false positives
↳ results are individually checked against query. ↳ low precision

→ Processing is expensive b/c of added lookup in special index, filtering & then standard inverted index.

SPELLING CORRECTIONS

- Forms of Spelling Corrections

① Isolated Term → Query = $(w_1, w_2 w_3)$ → correct single query term at a time.

→ check each word on its own for mis-spelling

→ will not catch typos resulting in correctly spelled word

eg. from → form

② Context sensitive (alternate is to use biword statistics in collection; use log of queries issued by users)

correct the typographical errors such as flew from or flew form

→ enumerate corrections of each query term, substitute the correction word & run query for max matching results → (Expensive b/c too many combinations)

→ fundamental premise → There is a lexicon (dictionary) from which the correcting spellings come from

↓ if not a lexicon based → find closest — If only 5 or few doc → we correct query & find closest.

(dictionary ke has word se query ke word ki relation baneyin & then give the closest)

Two basic principles:

① for mis-spelled query, choose the nearest one.

② when two correctly spelled queries are tied (or nearly tied), select one with more common.

grunt & grant → for grnt

↓ choose the one that occured more times in collection.

Techniques for addressing isolated term correction:

① Edit distance (levenshtein distance)

② Weighted Edit Distance

③ k-gram overlap

Day / Date: _____

Edit Distance:

eg. change cat → dog ; weight will be 3 → b/c 3 words change

fast → past ; 1 weight

Weighted Edit Distance:

B/w 2 strings in time $O(|s_1| * |s_2|)$ → $|s_1|$ → length of string s_i. Use dynamic Programming.

k-gram overlap:

fast	fa as st	$\frac{s_1 \cap s_2}{s_1 \cup s_2} = \frac{2}{4} = 0.5$ } jaccard coefficient
past	pa as st	

added distance sab se kam ho & prevelance max (people have used it in query at most)

added distance max = no. of letters in word.

1	}	check most frequent from this
2		from this
3		In k gram, k is usually 2 or 3.
4		jaccard value ↑ so words will appear at top
		jaccard value ↓ so words will appear at bottom

} ranking of documents is done on the basis of jaccard coefficient

PHONETIC CORRECTIONS

Mis-spellings that arise b/c the user types a query that sounds like the target term.

↳ main idea → to generate, for each term, a 'phonetic hash' so that similar sounding terms hash to the same value.

→ Algos for such phonetic hashing are commonly collectively known as 'Soundex Algos'

Soundex Algo is built on following scheme:

① Turn every term to be indexed into a 4-char. reduced form. Build an inverted index from these reduced forms to the original terms; call this the soundex index.

② Do the same with query terms.

③ When query calls for soundex match, search this soundex index.

Dazzle

Day / Date: _____

→ Issues in Spelling corrections

→ Spelling correction is computationally expensive

↓ only runned on docs which have result less than (close) to threshold.

→ Soundex

Class of heuristics to expand a query into phonetic equivalent.

↓ Soundex algo used (study it). → classic algo → not very useful for info retrieval.

↓ eg. Hermann maps to → H655

* Given a query we compute its soundex code & then retrieve all vocab term matching this soundex code from soundex index, before running the resulting query on the standard inverted index.

Context Sensitive ~~Analysis~~ Spelling corrections

↓ Hit Based Spelling Correction:

Suggest the alternatives that has lots of hits.

$$M = K \frac{1}{t^b}$$

$$cf_q \propto \frac{1}{q}$$

Dazzle

Day / Date: _____ Chapter 5

INDEX COMPRESSION (Chp 5)

→ WHY DO WE NEED COMPRESSION?

① Use less disk space (saves money)

② Increased used for caching; search sys uses some parts more frequently. Fit more info in main mem. (increased speed)

③ faster transfer of data from disk to memory. disk → memory

→ efficient decompression algos run fast; Premise: Decompression algos are fast (true).

→ Dictionary compression is important for effective IR sys esp. large scale.

→ Heap's Law: Estimating the no. of terms (simple relation b/w collection size & vocab size is linear in log-log)

→ Helps in estimate vocabulary size as a function of collection size)

$M = kT^b$ (vocab size M as a func collection size T)

Typical val for parameters: $30 \leq k \leq 100$, $b \approx 0.5$

→ parameter k is quite variable b/c vocab growth depends on collection (nature + how compressed)

→ case folding & stemming reduce growth rate of vocab; including num & spelling errors increases it.

→ Regardless of the value of parameters, Heaps law suggest ① dictionary size continues to ↑ increase with more docs in collections, rather than max vocab size being reached ② size of dictionary is quite large for large collections.

→ Zipf's Law: Modeling the distribution of terms

To understand how terms are distributed across doc; characterize the properties of & algo for compressing posting lists.

Zipf's law: If t_1 is the most common term in collection & t_2 is the next most common & so on, then collection freq cf_i of i^{th} most common term is proportional to $1/i$: $cf_i \propto 1/i$ ⟺ $\log(cf_i) = a/i$ (linear rel.)

• RCV1 Collection → used as benchmark data set; intuition is that freq decreases very rapidly with rank.

1 document has approx 200 token, n → 400, avg byte per token 6 (incl. spaces)

→ DICTIONARY COMPRESSION

Main goal → (*) To fit it in the main memory or atleast large portion of it to support high query throughput.

→ ① To be able to design limited search sys for limited hardware eg. phones bytes → one term of dict.

→ ② Fast startup time & having to share resources with other apps. → utilities

400 × 28 = 11.2 MB

① DICTIONARY AS A STRING

Sort vocab lexicographically & store it in an array of fixed width entries.

• 4 bytes → doc freq, 4 bytes → pointers to its posting list, 20 bytes → term

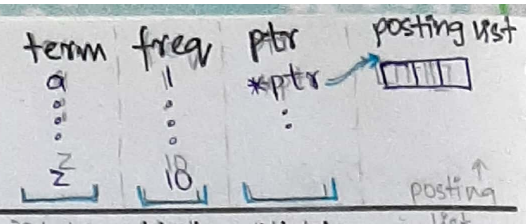
$M \times (20+4+4) = 400{,}000 \times 28 = 11.2$ MB for storing dictionary in this scheme.

for Reuters-RCV1

Dazzle

→ using fixed-width entries is wasteful
*avg term length in Eng = 8 char (12 wasted)
→ no way of storing terms of more than 20 char.

↓ Day / Date: _____



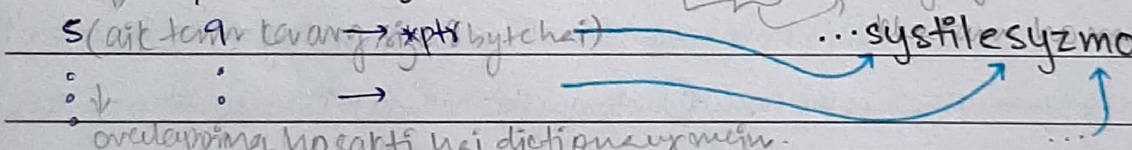
overcome These by

→ Storing dictionary terms as one long string of characters
* The pointer to next term is also used to mark the end of current term.
* saves us 60% space compared to fixed width

→ now also need to store term pointers.
→ term pointer resolves $400,000 \times 8 = 3.2 \times 10^6$ positions → $\log_2 3.2 \times 10^6 \approx 3$ bytes
↳ kahan ek word khatam aur doosra shuru.

so for Reuters-RCV1 → 4 bytes for freq and posting pointer, 3 byte for term pointer, 8 byte for term
→ $400000 \times (4+4+3+8) = 7.6$ MB.
→ reduced space by one third ie from 11.2 MB to 7.6 MB.

term | freq | posting ptr | term ptr



POSTING FILE COMPRESSION

→ 4 byte docID
→ offline indexing criteria
→ posting list > dictionary
→ since we'll pick one doc at a time, so we'll do increment of docID after scanning it for the query dictionary
• Alternate: Variable Length Encoding we'll do increment * docId after scanning
* uses fewer bit for encoding short gaps. ↖ ; short codes for small no.
→ for an economical representation of gaps
→ By wiki: maps source symbol to a variable no. of bits.
→ difference b/w docID will be place so that the gap to each next docID will be (docID + difference b/w 2 docIDs)
* Avg gap for a term is G, we want to use $\sim \log_2 G$ bits/gap entry

INDEX COMPRESSION → Improves time & space efficiency of indexes by ↓ amount of disk needed
↑ amount of info that can be kept in cache & speeding up data transfer from disk to memory
(only 4% size of total size of collection & 10-15% of total size of text in collection)

(search parse)

Day / Date: _____

INDEX CONSTRUCTION (Chp 4)

→ ~~How to efficiently make index~~

→ HARDWARE BASICS

While building an IR sys, many decisions are based on characteristics of comp hardware on which sys runs.

① Access to data in memory > Access to data on disk.

② When W/R on disk, it takes a while for disk head to move to the part of the disk where data is located. This is called seek time; avg is 5ms. No data being transferred during seek so to maximize data transfer rate, chunks of data that will be read together should be stored contingously on disk.

③ Transferring blocks of large data is easier than transferring small chunks; buffer → where block is being R/W

④ Fault tolerance is expensive. RAID is used to avoid fault tolerance.

⑤ processor is available to process data during disk I/O so we use it to speed up data transfer by storing compressed data on disk.

→ • Basic Inverted Index (already studied)

• Block Sorted Based Index (BSBI)

→ Ak doc read krke uske word & docID buffer pe dalay hain & when buffer = full, we put it on the disk after sorting. So we have blocks of sorted words. Then we merge it with help of pointers in such a way that repitition doesn't take place.

↑ better than previous process

→ use when data set cannot be kept in memory. Block by block we can easily fetch query result from the disk. ous process.

→ use when data set to memory mein nai rakh saktay.

• Single-Pass-In-Memory-Indexing

Key Idea ~ Generate seperate dictionaries

Agar koi word already dictionary mein hai tou bas usko posting mein ++ krenge.

BSBI se faster b/c sort nai (hashing kr rahay)

Agar koi word already present hai tou bas posting LIST mein ++ krenge bas faster than BSBI b/c sort nai kr rahay (hashing kr rahay)

Dazzle

Day / Date: ____________

- Distributed Index (Search Engine do this)

→ Maintain a master machine

① Parallel Task ② Parser ③ Index
↓
Blocks aik master machine bnarani aawo machine sab blocks indexing machine ko detii hai in a sorted way after taking chunks back from indexing machine.

→ all decisions are taken by master machine.

Day/ Date:

Chp 6

Scoring, Term Weighting & Vector Space Model (VSM)

Limitations of Boolean Model

↓ takes a lot of skills to come up with a query that produces a manageable no. of hits.

↓ and gives too few or too many documents.

Ranked Retrieval

→ Result is showed in decreasing order of ranking

→ If score is equal, the doc are of both equal importance

$f = q, d_i) = \{d_1, \dots d_n\}$ ↑ → in precomputed way

query — collection of doc — order — permutation

no. of result

→ and we set make ke kas kitna show karna hai

→ Free text queries :

→ function of score - ranking

max → if identical (not same)

min → if totally different

Score is based on [0, 1] → totally same

↓ totally different

Query - Matching Document

(→ freq. matters)

Assign a score to a query/doc pair.

Jaccard Query → $\frac{|D_i| \cap |q|}{D_i \cup q}$

$w_1^2\ w_1^1$ — multiset

$w_p^1\ w_2^1$) every doc is considered as a bag of words

The more freq the query term, the higher the score

Sim(d_i, q) $q = w_i\ w_p$

$R = \frac{5}{8}$

w_i	w_p w_p
w_p w_2	w_p
w_q	w_q

Dazzle

Day / Date: _____

→ Limitation

→ favoured for smaller doc ; biased for shorter doc, shorter doc ā top.

Parametric Search / Meta Search

→ common in digital libraries

↓ provide search on basis of parameters.

→ Weighted Zone Scoring

$\sum_{i}^{n} g_i = 1$ → sum of all scores of zones.

→ zone is abstract, can be on any basis

$s(Q, D_i) = \sum_{i} g_i \times score(q, D)$

↓ (weight of a zone)

↳ we try to learn weight by supervised learning mechanism

→ freq. plays a role but rā in this algo

(partial ordering) (search)

(+ ranking + arrangement of result)

Train → $f(X)_T = y$

Test → $f(X_T) = y \rightarrow y$

Learning weights for Zone Scoring

query	user doc	rank
		0
		1
q_1		0

→ explicit relevance feedback

→ use ko query aur doc dikha ke relevant ki ranking krwayein ge

→ multiple users to overcome biasness.

(supervised learning)

dataset | $x \rightarrow y$ | humanly annotated → log use judge

Training data
Testing data
Labelled supervised data

$x_i = \{f_1 \cdots f_n\} = R/R$

Dazzle

Day / Date: _____

Learning weights for scoring zone.

$Q = t$

$D = d_i = [\underbrace{t_1}_{\text{title}} \quad \underbrace{t_1}_{\text{body}}]$

$S(q, d_i) = g\, S_t(t, d_i) + (1-g) S_b(t, d_i)$ ← how to determine g (constant)

↓

$\varepsilon(g, \phi_j) = (r(d_j, q_j) - score(d_j, q_j))^2$

→ (incorporate all conditions)

↓

$\sum_j \varepsilon(g, \phi_j)$ → need to sum all

↓ jitna kam error (min. error) hoga utna optimized sol.

⋮

optimal value of g → $\dfrac{n_{10r} + n_{01r}}{n_{10r} + n_{10n} + n_{01r} + n_{01n}}$

n_{10} / n_{01} → any term

n_{10} → like mein prese. body mein nai

r → relevant

n → non relevant

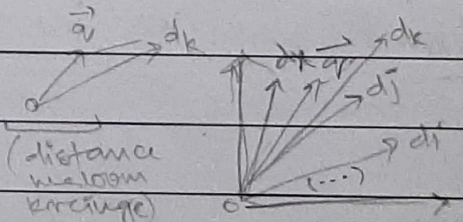
(in subscript)

Bag of words

Score – CommonTerms in doc & query / total terms (use jaccard query)

Issues –

① doc. size affects answer



$\vec{a} = c_1\hat{x} + c_2\hat{y} + c_3\hat{z}$

Alg.

Model for repr. text doc as vectors of identified unique.

Every term considered as a dimension.;

terms bi binary in dictionary, is in order → dictionary is of d dimension. & d dimension hyper space.

Used in info filtering, info retrieval, index

Dazzle

Day / Date: _____

Vector representation of Query

- Query ($\vec{q}$) is also represented similar to doc.
- Use distance b/w vector as a distance of closeness
- V-dimensional vector is also used for query only the term present in query are non-zero vector for query.

$$\cos(d_i, q) = \frac{\vec{d_i} \cdot \vec{q}}{|d_i| \cdot |q|}$$

[angle b/w q & doc]

cosine similarity

① if angle close, doc is closer (similar)
vector distance reject b/c sensitive

② har doc ko uni vector se represent

$$\vec{v} = |v| \cdot \hat{n}$$

$$\hat{n} = \frac{\vec{u}}{|u|}$$

→ $\cos(\theta) \cong \theta$

→ jo term query mein aa rahi hai, sirf uske liye calculate karenge.

Term Frequency

→ How many times a term has appear in a doc.

$t_{(d,t)} \Rightarrow \text{love}_{(1,3)}$

doc no. ↙ ↑ freq

Sublinear 'tf' scaling

b/c agr words bhi bahi toh length would increase at the speed of (x freq)

→ Term freq. can be scaled in different ways (tf)

* log linear → $1 + \log(tf_{t,d})$
* fragmented → weighting scheme → $0.5 + \frac{0.5 \times tf_{t,d}}{\max_t(tf_{t,d})}$

Dazzle

Day / Date: _____

* Maximum freq. normalisation (har term ane wala max)

Doc freq.

→ collection ke kitnay doc mein aik term aayi hai

df = 3 ⟶ 3 doc mein aayi

(taka. ~~inverse~~ doc. freq)

① Scaling the df : $idf_t = \log \frac{N}{df_t}$ ⟶ no term jo kam doc mein ayi hai will have ~~highest val~~ discriminating power.

↳ agr koi term sab doc mein hai so no advant. in retrieval.

② Augmented — $\max\{0, \log \frac{N - df_t}{df_t}\}$ { → saray doc ko such weight de takay they can partially get score

have ↓ nai dimension doc ki mein add kr rahi hai.

Best weighting Scheme

= (term freq) × (inverse doc freq)

$= tf - idf_{t,d} = tf_{t,d} \times idf_t$

assigns a weight to term

① highest when t occurs many times within a small no. of doc

② lower when term occurs fewer times in a doc or occurs in many doc

③ lowest when term occurs in almost all doc

3 imp properties

If we are keeping doc freq = 1 so means that we have not considered any doc's freq.

SMART SYSTEM — 60 possible config.

(ddd.qqq)

- Advantages of VSM
- Disadvantages of VSM

Dazzle

Day / Date: _____ singular similarity

Ex: 1 → 6.10

→ doc x idf_t

{ information retrieval. }

simple

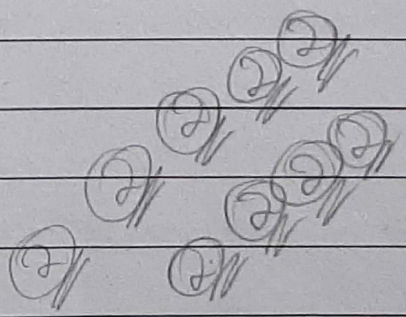
stable + efficient stable sort

n log n

a	doc1	after	a	doc1
b	doc1	⇒		doc2
c	doc1	sort		doc3
⋮	⋮		⋮	⋮
z	doc2		z	doc2. doc4
⋮	⋮			

↓

efficient way is to pick a doc BUT by after buffer is filled shift to disk in blocks



Dazzle

Day / Date: _____

x – BOOK NOTES – x

(chp1)

→ Building an Inverted Index

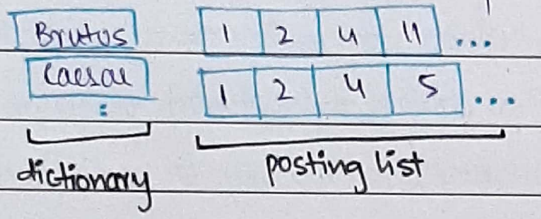
→ (to gain speed benefits of indexing at retrieval-time, build index in advance:)

① Collect docs to be indexed. Friends, countrymen, Roman | So let it be ceased...

② Tokenize text by turning each doc in list of tokens. Friends | countrymen | Roman | So | let | it ...

③ Do linguistics preprocessing by producing a list of normalized tokens. Friend | Roman | countryman ..

④ Index the doc that each terms occurs in by creating an inverted indexing consisting of dictionary & posting list.



* each doc has a unique docID.
* dictionary also stores doc freq = length of posting list.
↓
increases efficiency of search engine @ query time + used for ranked retrieval model

→ This is then later sorted & grouped to make an index.
→ The dictionary stores the terms & has a pointer to the posting list for each term.

→ Inverted Index is the most efficient structure for supporting adhoc text search.

* We pay for storage of both; dictionary kept in memory while posting list in disk (since large)

what data structure should be used for posting list?

① Singly linked list ② variable length array

→ Processing Boolean Queries

→ Brutus AND Calpurnia

① locate Brutus from dictionary
② Retrive its posting list
③ locate calpurnia in dictionary
④ Retrieve its posting list
⑤ Intersect the two posting list

Brutus → 1 → 2 → 4 → 11 → 31 → 45 → 173 → 174

Calpurnia → 2 → 31 → 54 → 101

Intersection → 2 → 31

*Merge Algo

```
Intersect(p1, p2) {
answer ← ()
while p1 ≠ NIL and p2 ≠ NIL
do if docID(p1) = docID(p2)
   then ADD(answer, docID(p1))
        p1 ← next(p1)
        p2 ← next(p2)
   else if docID(p1) < docID(p2)
        then p1 ← next(p1)
        else p2 ← next(p2)
return answer }
```

Dazzle

Day / Date: _____

(Chp 3)

→ SPELLING CORRECTIONS

① EDIT DISTANCE

2 character string s_1 & s_2 ; edit distance b/w them is minimum no. of edit operations required to transform s_1 into s_2.

common edit operations allowed :

① insert a character in string ② Delete a character ③ replace a character with another.

edit distance aka levenshtein distance

eg. cat & dog ; distance is 3.

→ Weighted Edit Distance

→ B/w two strings in time $O(|s_1| \times |s_2|)$; use dynamic programming.

→ char. in s_1 & s_2 are in array form. Algo fill the (integer) entries in a matrix m whose dimensions equal the lengths of two strings who edit distance is being computed :

→ (i, j) entry of matrix will hold the strings consisting of first i char of s_1 & first j char of s_2.

EDIT DISTANCE(s_1, s_2)

int m[i, j] = 0

for i ← 1 to $|s_1|$

do m[i, 0] = i

for j ← 1 to $|s_2|$

do m[0, j] = j

for i ← 1 to $|s_1|$

do for j ← 1 to $|s_2|$

do m[i, j] = min { min [i-1, j-1] + if ($s_1[i] = s_2[j]$) then 0 else 1 if

m[i-1, j] + 1,

m[i, j-1] + 1 }

return m[$|s_1|$, $|s_2|$]

→ The spelling correction demands more than computing edit distance.

↓

this exhaustive search is expensive.

Day / Date: _____

② K-gram Indexes

→ to further limit the set of vocab term for which we compute edit distances to query term.

↓ K-gram: to assist with retrieving vocab terms with low edit distance to query q.

x — x

Chp 5.4

→ BLOCKED SORT-BASED INDEXING (excellent scaling properties, but needs a DS for mapping terms to termIDs. For large collection, DS doesn't fit in memory)

→ to make index construction more efficient, represent terms as termIDs (where each is unique sno.)

→ build mapping from terms to termIDs on the fly while we are processing the collection.

→ Reuter-RCV1 collection as our model collection — 1GB text → 800,000 documents

BSBI:

(i) Segments the collection into parts of equal size.

(ii) Sorts the term ID - doc ID pairs of each pair in memory.

(iii) stores intermediate sorted results on disk

(iv) merges all intermediate results into final index

```
BSBIndex()
n ← 0
while (all doc have not been processed)
do n ← n+1
    block ← ParseNextBlock()
    BSBI-Invert(block)
    WriteBlockToDisk(block, fn)
MergeBlocks(f1, ... fn; fmerged)
```

① parse doc in termID-docID pairs & accumulates the pair in memory until a block of fixed size is full.

② we choose block size to fit comfortably into memory to permit a fast in-mem sort.

③ Blocks are then inverted & written to disk; Inversion has 2 steps

(i) We sort termID-docID pairs (ii) We collect all termID-docID pairs with same termID into a posting list where posting is simply a docID.

④ The result is then written to the disk

→ With RCV1 we end up with 10 block, each an inverted index of one pair of the collection.

⑤ Finally, we merge 10 blocks into 1 large merged index.

TIME COMPLEXITY → $O(T \cdot \log T)$

Dazzle

Day / Date: _____

→ SINGLE-PASS IN-MEMORY INDEXING

More scalable alternative is SPIMI. → uses terms instead of termIDs, writes each block's dictionary to disk & then starts a new dictionary for the next block.

→ SPIMI can index collections of any size as long as there is enough disk space available.

→ SPIMI-Invert is called repeatedly on token stream until the entire collection has been processed.

→ Tokens are processed one by one during each successive call of SPIMI-Invert.

→ When a term occurs for first time → added to dictionary (best as hash), and a new posting list created (line 6)
→ the call in line 7 returns this posting list for subsequent occurrences of the term.

→ Difference b/w BSBI & SPIMI → SPIMI adds a posting directly to its posting list (line 10).
Each posting list is dynamic & is immediately available to collect postings.

Advantages:

① faster b/c there is no sorting required

② Saves memory b/c we keep track of ~~the~~ term a postings list belongs to so termID doesn't need to be stored.
→ whole lot process is more efficient

Three components of SPIMI:

① Constructing new dic. DS for each block
② Eliminates expensive sorting
③ Compression; posting & dictionary can be stored compactly on disk if we employ compression. (Increase efficiency)

TIME COMPLEXITY: O(T) b/c no sorting & almost all operations are linear.

Dazzle

Day / Date: _____

→ DISTRIBUTED INDEXING

Web scale search engine cannot build index on a single server — large data.

→ The result of construction process is a distributed index partitioned across several machines — either acc. to term or acc. to doc — we describe for acc. to term

→ Application MapReduce → designed for large comp clusters.

→ A master machine (node) directs the process of assigning task to individual nodes

> map & reduce phases of MapReduce split up computing job into chunks that standard machine can process in short time.

① Input data (collection of web pgs) split into n splits where the size of the split is chosen to ensure that the work can be distributed evenly & efficiently (16 or 64 MB size)

> Splits are not pre-assigned; they are assigned by master node on an ongoing basis.

→ > As a machine finishes processing one split, it is assigned another

→ > If a machines dies or become laggard, the split it is working on is assigned to another machine

>> 'key-value pair' is →: term → termID mapping.

Map phase:

→ Mapping splits of input data to key-val pair. Same parsing task as BSBI & SPIMI, so called parser machines are

→ Each parser writes its output to local intermediate files, the segments files | a-f | g-p | q-z |

Reduce phase:

> we want all values for a given key to be stored close together → read & processed quickly.

↑ Achieved by partitioning key into j term partitions & having parser write key-val for each term partition into a seperate segment file

j=3

> Collecting all val (docID) for a given key (termID) into one list is task of inverters.

• Master assigns each term partition to a different inverter.

> same machine can be parser in map phase & inverter in reduce phase; depends on if idle.

→ Each parser writes its segment file to its local disk; minimize amount of traffic in indexing.

Conclusion

→ simple & Robust framework

Dazzle

Day / Date: _____

→ DYNAMIC INDEXING

Daz

q ~~d~~ ~~d_1~~ $\frac{d_1 q_1 + d_2 q_2 +}{\sqrt{(d_1)^2 + (d_2)^2}}$

Day/Date: Someone! & dot

vocab = {w1, w2 w3 w4 w5}

$d_1 = \{w_2\ w_3\ w_1\ w_3\ w_2\ w_1\ w_5\} = 5$

$d_2 = \{w_3 w_2 w_1 w_4\}$

$d_3 = \{w_1, w_3 w_2\}$

q ~~d4~~ $= \{w_1 w_2 w_1\} = 2$

Similar. ⇒ Simil df → Similar

{auto, company, moto}

q = tf * idf

$d_1 = \{$auto

$d_2 =$ company moto

$d_1 \langle 1, 0, 0 \rangle$

~~$d_2 \langle 0, 1, 1 \rangle$~~

$d_1 \langle 1, 2, 2, 1, 1 \rangle$

$d_2 \langle 1, 1, 1, 1, 0 \rangle$

$d_3 \langle 1, 1, 1, 0, 0 \rangle$

$q \langle 2, 1, 0, 0, 0 \rangle$

$D_1 \langle 0, -0.24, -0.24, 0.18, 0.47 \rangle$

$sim(d, q) = \frac{\vec{V}(d) \cdot \vec{V}(q)}{|\vec{V}(d)||\vec{V}(q)|}$

N = 3

tf idf $= \log \frac{N}{df}$ $\log \frac{N}{df}$

ayesha. ⇒

seriously

$\log \frac{3}{3} = \log 1 = 0$

~~D1 D2~~	df	idf
w1	~~3~~	0
w2	4	-0.12
w3	3	-0.12
w4	2	0.18
w5	1	0.47

monday

tf*idf

summers.

sunday!

Ayesha.

ayesha.

ayesha J(C)

ayesha.

$JC(d_1, q) = \frac{d_1 \cap q}{d_1 \cup q} = \frac{2}{5}$

Dazzle

Day / Date: _____

$d_1 = \{w_1\ w_2\ w_3\ w_4\ w_3\ w_5\}$

$d_2 = \{w_1\ w_2\ w_6\ w_4\}$

$d_3 = \{w_2\ w_5\ w_6\ w_7\}$

$q = \{w_5\ w_4\ w_7\}$

$N = 3$

$TF = 1 + \log(tf_{t,d})$

$idf = \log\left(\frac{N}{df_t}\right)$

$tf * idf$

EF*idF-D1

	idf	tf-D1	tf-D2	tf-D3	tf-q	
w_1	0.176	1	1	0	0	0.176
w_2	0	1	1	0	0	
w_3	4.77	1.301	0	1	0	
w_4	0.176	1	1	0	1	$\frac{\vec{V}(D_2) \cdot \vec{V}(q)}{\lvert\vec{V}(D_2)\rvert \cdot \lvert\vec{V}(q)\rvert}$
w_5	0.176	1	0	1	1	
w_6	0.176	0	1	1	0	
w_7	0.477	0	0	1	1	$\frac{\vec{V}(D_2) \cdot \vec{V}(q)}{\lvert\vec{V}(D_2)\rvert \cdot \lvert\vec{V}(q)\rvert}$

	$sim(D_1, q)$	$sim(D_2, q)$	$sim(D_3, q)$
w_1			$D_2 q +$
w_2			
w_3			
w_4			
w_5			
w_6			
w_7			

$$sim(d_1, q) = \frac{(0.176)^2 + (0.176)^2}{\left(\sqrt{(0.176)^2 + (0.62)^2 + (0.176)^2 + (0.176)^2}\right)}$$

Dazzle

Day / Date: _____

EVALUATION IN INFORMATION RETRIEVAL (Chp 8)

* Intro:

8.1 — Information Retrieval System Evaluation

To measure adhoc info retrieval effectiveness, we need test collection consisting of 3 things:

① A doc collection. ② A test suite of info needs, expressible as queries.

③ A set of relevance judgements, standardly a binary assessment of either relevant/non-rel. for each query-doc pair.

> Wrt user info need, a doc in collection is given a binary collection as either rel/nonrel. This decision is referred as gold standard or ground truth judgement of relevance.

> Relevance is assessed relative to an info need, not a query.

→ Many sys contains weights (known as parameters) which can be adjusted to tune sys performance

↓ It is wrong to ~~test~~ report ~~those~~ results on a test collection which were obtained by tuning these parameters to maximize performance on collection; we should have development test collection to tune the parameters dev test collection. The tester run those weights on test collection & gives a result ~~which~~ as unbiased estimate of performance.

~~8.2 — Standard Test Collections~~

→ Intro: Different IR Models:

Many retrieval models are present, the best component for (Ranking func → dot-prod, cosine), Term selection → stopword removal, stemming) & (term weighting → TF, TF-IDF).

→ Difficulty in IR models:

Effectiveness is related to relevancy of retrieved items; typically binary.

8.2 — Standard Test Collections

→ Cranfield → TREC (Text Retrieval Conference) → GOV2 → NTCIR

→ CLEF → Reuters → 20 News group

(read details from book 154)

Dazzle

Day / Date: _____

8.3 — Evaluation of unranked retrieval sets

> Most basic & frequent IR effectiveness measures are: ① precision ② recall

Precision — fraction of retrieved doc that are relevant.

$$P = \frac{\#(\text{relevant items retrieved})}{\#(\text{retrieved items})} = P(\text{relevant} \mid \text{retrieved})$$

Recall — fraction of relevant doc that are retrieved

$$R = \frac{\#(\text{relevant items retrieved})}{\#(\text{relevant items})} = P(\text{retrieved} \mid \text{relevant})$$

So, $P = tp/(tp+fp)$ where, tp = rel. + retrieved, fp = Non-rel + retrieved

& $R = tp/(tp+fn)$ where, fn = not retrieved + rel.

* since result of a query set is unranked (flat), we use P & R to evaluate.

	Rel (relevant)	NotRel		
Ret	Ret_{rel}	Ret_{notrel}	$P \in [0,1]$ $P = Ret_{rel}/Retrieved$	$Ret = Ret_{rel} + Ret_{notrel}$
NotRet (retrieved)	$NotRet_{rel}$	$NotRet_{notrel}$	$R \in [0,1]$ $R = Ret_{rel}/Retrieved$	$NotRet = NotRet_{rel} + NotRet_{notrel}$

→ an alternative to judge an IR sys by its accuracy is fraction of its collection that are correct.

> Accuracy is not a good measure for IR prob b/c data is extremely skewed (many times)

> Precision & recall concentrate the evaluation on the return of true +ve.

Precision vs. Recall

Sys makes 2 types of errors:

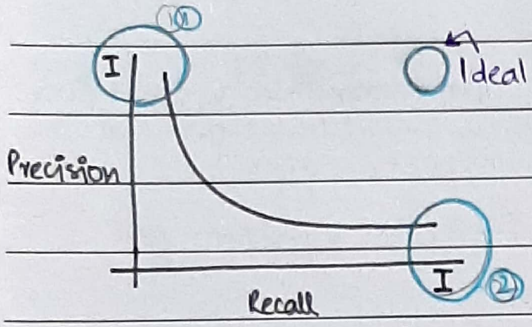
false +ve → retrieves a doc that is non-relevant (should not have been retrieved)

false -ve → fails to retrieve a doc that is relevant (" have been retrieved)

Dazzle

Day / Date: _____

> false +ve affects → precision ; false -ve affects → recall



① Retrieves relevant doc but misses many useful too

② Retrieves most relevant doc but includes lots of junk

> Precision & Recall are inversely proportional.

• In some cases, precision matters more than recall & vice versa.

> F-measure — A single measure which trades off precision vs. recall.

F-measure is the weighted harmonic mean of precision & recall.

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R} \quad \text{where } \beta^2 = \frac{1-\alpha}{\alpha} \quad \text{where } \alpha \in [0,1] \text{ & } \beta \in [0,\infty]$$

→ Default F measure equal weights P & Recall ; $\alpha = 1/2$ or $\beta = 1$ if $F_{\beta=1} = \frac{2PR}{P+R}$.

→ However if $\beta < 1$ → emphasize P & if $\beta > 1$ → emphasizes R.

→ recall, precision & f-measure are b/w 0 & 1.

Precision Critical Tasks

→ Time matters alot, Tolerance to missed doc, Redundant: many equal info resources

Eg: Web search & demand is very high. General Optimization

Recall Critical Tasks

→ Time matters less, Non-tolerance to missed doc, Less Redundant info: only one or few resource

Eg. Legal/patent search & moderate demand. Specific optimization.

> $F_{\beta=1}$ is called harmonic mean of P & R.

↓ Harmonic mean is advantageous to use b/c it is more closer to min(P, R), whereas arithmetic mean is closer to the highest of two values of P & R.

Day / Date: _____

Fallout Rate

Problems with Precision & Recall:

① → No. of DNR not taken into account. ② → Recall undefined → no relevant doc in collection

③ → Precision undefined → no doc is retrieved.

$$\text{Fallout} = \frac{\text{no. of DNR retrieved}}{\text{total no. of DNR in collection}}$$

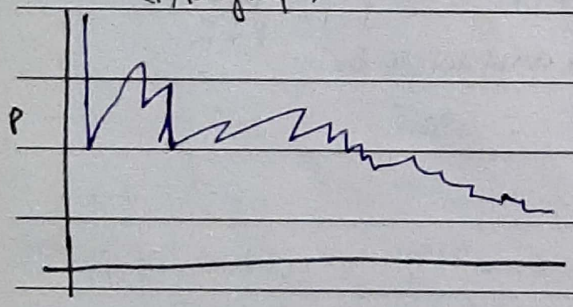
8.4 Evaluation of ranked retrieval results

→ P, R & F measure are set based measures & computed using unordered set of doc.

→ In a ranked retrieval context, appropriate sets of retrieved doc are naturally given by top k retrieved doc.

→ For each such set, precision & recall values can be plotted to give a precision-recall curve.

(P/R graph) (used compute ranked result)



→ saw tooth shape

→ If $(k+1)^{th}$ doc retrieved as non-rel. then recall is same as for the top k doc, but precision dropped.

→ If relevant, then both P & R increase & curve jags up and to the right.

Interpolated precision p_{interp} is used to remove jiggles; p_{interp} at a certain R level r is defined as the highest precision found for any recall level $r' \geq r$:

$$p_{interp}(r) = \max_{r' \geq r} p(r')$$

n	doc #	relevant	#DR=6		
1	588	x	→ R = 1/6 = 0.176	; P = 1/1 = 1	
2	589	x	→ R = 2/6 = 0.333	; P = 2/2 = 1	
3	576				
4	590	x	→ R = 3/6 = 0.5	; P = 3/4 = 0.75	Since missing one rel. doc, never reach 100% recall
5	620				
6	990	x	→ R = 4/6 = 0.667	; P = 4/6 = 0.667	
⋮	⋮	⋮			
13/14	772	x	→ R = 5/6 = 0.833	; P = 5/13 = 0.38	

Dazzle

Day / Date: _____

Average Precision

▨ → relevant docs → total = 6.

▨ ☐ ▨ ▨ ▨ ▨ ☐ ☐ ☐ ▨

R 0.17 0.17 0.33 0.5 0.67 0.83 0.83 0.83 0.83 1.0
P 1.0 0.5 0.67 0.75 0.8 0.83 0.71 0.63 0.56 0.6

~~avg~~ ranking = (1.0 + 0.67 + 0.75 + 0.8 + 0.83 + 0.6)/6 = 0.78

Mean Average Precision (MAP)

> provides single-figure measure of quality across recall levels.

> MAP shows good discrimination & stability; good for web search

→ for single info need, avg precision is the avg of precision val. obtained for the set of top k docs existing after each rel. doc is retrieved, and this val. is then ~~obtained~~ averaged over info needs.

> If rel. doc never gets retrieved, we assume precision corresponding to that rel doc = 0.

> MAP is macro-averaging - each query counts equally.

> MAP assumes user is interested in finding rel. doc for each query.

> MAP requires many relevance judgements in text collection.

> Has adv. of not requiring any size of doc set; drawback: least stable, not good avg.

Precision at k: measuring P at low fixed levels of retrieved result.

R-Precision → requires set of known relevant (Dr) doc returned from which we can calculate the precision of top Dr.

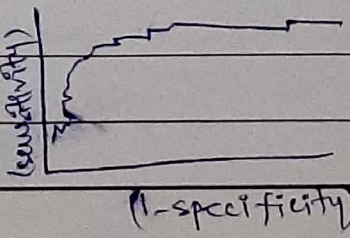
Break-even point → when R Precision & the recall are same r/|Dr| ~~mean~~; r → relevant.

ROC - curve - Reciever Operating characteristics

> Plots true positive rate or sensitivity against false +ve rate or (1 - specificity); sensitivity = recall

> false +ve rate is given by fp/(fp+tn)

roc →



Dazzle

Day / Date: _____

Kappa Statistics (from slide 14 W7)

$K = (P(A) - P(E)) / (1 - P(E))$; where $P(A)$ = Observed proportion of the times judges agreed (YES|YES) (NO|NO)

$P(A) = (YES|YES + NO|NO) / total$.

$PE = P(nonrelevant)^2 + P(relevant)^2$.
↳ (proportion of times the judges would be expected to agree by chance)

- K = b/w 0-1 ; 0.6-0.8 → fair agreement ; 0.8 → good agreement

(consensus pe agree krtay hain judges)

→ A common measure for agreement b/w judges is the kappa statistics.

→ Cummulative Gain (CG)

Sum of graded relevance val of all results in a search result list; judges score the doc.

Ex: for query → q, D_1 D_2 D_3 D_4 D_5 D_6 are relevant docs.

3 2 3 0 1 2 ⇒ CG = 11

> Only change in d bring a change in CG ;

> Drawback – Changing of order doesn't affect CG measure.

→ Discount CG (DCG)

DCG is total gain accumulated at a particular rank p: $DCG_p = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2 i}$ or $\sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log(1+i)}$

→ emphasizes on retrieving highly relevant docs.

> if rel = 1 ; order = log(1+1) → DCG = 3 (slide 16 of Week 7 – solved example)

→ Normalized DCG at rank n

NDCG at rank n by DCG value at rank n of the ideal ranking

Ideal order of result = sort in decreasing order of doc relevancy.

Useful → for contrasting queries with varying no. of relevant results; used in web search.

$$nDCG = \frac{DCG}{Ideal\ DCG}$$

(see slide 18 for more solved sol — Week 7)

Dazzle

Day / Date: _____

8.5 – Assessing Relevance

> To properly evaluate sys, given infoneeds & doc needs, you need to collect relevance assessments
> Standard approach is pooling → relevance is assessed over a subset of collection that is formed from the top k doc returns by a no. of different IR sys

8.6 – A broader perspective: System quality & user utility

Systems that allow quantitative evaluation & the issue of user utility.

– System Issues

Several benchmarks to rate an info retrieval system beyond its retrieval quality, they are:

① How fast does it index, how many doc/hr it index over doc lengths? } (measurable & can quantify size or speed)
② How fast does it search ie what is latency as a func. of index size?
③ How expressive is it query lang. & how fast is it on complex queries?
④ How large is doc collection?

– User Utility

- Way of quantifying aggregate user happiness based on relevance, speed & use interf. of sys.
- Take survey from user (feedback); if cust → happy, it will use sys again. (return to same engine)

– Refining a deployed system

- A/B testing

> Precisely one thing is changed b/w the current sys & proposed sys, and a small proportion of traffic (1–10% users) is randomly directed to the variant sys, while most users use current sys.
> click through log analysis or click stream mining. To see whether user like it or not.
> The basis of A/B testing is running a bunch of single variable tests (either in seq or in parallel): for each test only one parameter is varied from the control (live current sys)
> A/B finds it challenging to which part it should evaluate.

Dazzle

Day / Date: _____

8.7 — Result Snippets

> Search Snippet is useful for reviewing the search results.

> Two kind of summaries:

Static — which are always the same regardless of query ← drawback

— fast computing → (advantage)

Dynamic — (aka Query dependant) which are customized acc. to the user's info need as deduced from a query. ~~Dynamic~~ Dynamic summaries attempt to explain why a particular doc was retrieved for the query at hand.

* where the word appears in doc, it takes 100 words before & after the doc and processes a summary of it.

- Q dependant → user explains query very well.
- Advantage → give users full idea of doc.
- Drawback → time consuming eg. google.

→ ~~Keyword-in-context (KWIC) snippets~~

Dynamic summaries display one or more "windows" on doc, aiming to present the pieces that have the most utility to user in evaluating doc w.r.t users need of info

→ usually these windows contains one or several of q terms, and so are referred to as keyword-in-context (KWIC) snippets.

Note: find everything relevant — high recall

~~find~~ only retrieve what is relevant — high precision.

Day / Date: _____

Relevance Feedback & Query Expansion (Chp 9)

- **Problem with IR system**

→ Also known as 'synonymy' → has an impact on recall of most IR systems.

* users address this prob by refining query themselves ; Query refinement / Expansion.

→ Method for tackling is → ① Global Methods ② Local Methods

- **Global Method:** Reformulating query terms independant of the query & result returned from the query. Changes with cause it to match other semantically similar terms.
  - Using Thesaurus , via automatic Thesaurus generation , ~~sp~~ spelling corrections
  - ~~search engine~~ indexing data

- **Local Method:** Adjust query relative to a doc that initially appear to match the query.
  - Relevance Feedback → involve user feedback (to get) from results return for a query.
  - Pseudo Relevance Feedback → blind relevance feedback ; Global (Indirect) R feedback.
    → to info user end pc how.

9.1 — Relevance Feedback & Pseudo Relevance Feedback.

- Idea of relevance feedback → involve users in the retrieval process as to improve final result set
  → in particular, user gives rf on initial set of docs ; Rf can go through a no. of iterations.

Basic procedure: ① → user issues a simple/short query. ② → sys returns initial set of retrieval result
③ → user marks doc as R or NR. ④ → sys computes a better representation of info need based on user feedback. ⑤ → sys displays a revised set of retrieval result.

* we assume user is not able to formulate query properly so we involve him to get perfect result.

9.1.1 — Rocchio Algorithm

→ classic algo for implementing relevance feedback ; model of a way to incorporate rf info in vector space model (VSM).

→ compute new query in VS.

→ need to find query vector, $\vec{q}$ that maximizes similarity with R doc & min. similarity with NR.

centroid → vs ka common point.

$\vec{\mu}(\vec{c}) = \frac{1}{|c|} \sum_{d \in c} \vec{d}$ where, c = set of docs.

Day / Date: _____ (sum of all vectors / no. of doc marked R by user)

$\vec{q}$ = query vector

C_r = set of relevant docs

C_{nr} = set of NR docs

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

under cosine similarity, $\vec{q}_{opt} = \frac{1}{|C_r|} \sum_{d_j \in C_r} \vec{d_j} - \frac{1}{|C_{nr}|} \sum_{d_j \in C_{nr}} \vec{d_j}$

$\vec{\mu}(c) = \frac{1}{|c|} \sum_{d \in c} \vec{d}$ (c → set of docs)

← optimal query is vector difference b/w the centroids of the R & NR doc.

(not precise cause we dk the full set of R doc

$\alpha > 0$ → user have good judgement of initial query

$D_r \geq 5$ - marked by user then $\beta > 0$ & $\beta > \gamma$

α → based on initial q

β → w of D_r, γ = w of D_{NR}

-ve weight → if $\beta < \gamma$ ↳ (term ignored)

push towards +ve

Rocchio Algo → $\vec{q_m} = \alpha \vec{q_0} + \beta \frac{1}{|D_r|} \sum_{d_j \in D_r} \vec{d_j} - \gamma \frac{1}{|D_{nr}|} \sum_{d_j \in D_{nr}} \vec{d_j}$

pulls away from -ve

where, $\vec{d}_m$ = modified query, $\vec{q_0}$ = original query ; α, β, γ → weights of terms

(cause user gave ambiguous)

→ query move toward R doc & away from NR.

→ tradeoff α vs. β/γ = if we have a lot of judged docs, we'd want a higher β & γ.

→ Negative term weights are ignored ie. term weight is set to 0.

→ Rf can improve both precision & recall. ; most useful for increasing recall in certain situations

9.1.2 Probabilistic Relevance feedback

Rather than reweighing query in VSM, we can use NaiveBayes on the basis of R & NR doc.

$P(x_t = 1 | R = 1) = |VR_t| / |VR|$

$P(x_t = 1 | R = 0) = (df_t - |VR_t|) / (N - |VR|)$

R → relevant or not

N → total no. of docs

df_t → no that contains t.

VR → set of known relevant docs

x — x

Explicit fd → user ~~doesn't~~ know that he is given feedback

Implicit / indirect fd → user doesn't know ; less reliable but more useful than pseudo fd.

Dazzle

Day / Date: _____

9.1.3 – When does Relevance Feedback Works?

Success of Rf depends on few assumptions:

- **A1**: User has sufficient knowledge for initial query (user formulated a good q/c. $\alpha > 0$)
- **A2**: Distribution of term in D_r is similar ie. Relevance prototypes are well behaved
  - Term distr. in NR doc will be different from those in D_R.

→ either all doc R are tightly clustered around a single prototype or there is a vocab overlap.

Violations

A(1): User doesn't have sufficient initial knowledge.

eg. (i) Misspellings (Brittany Speers) (ii) cross lang. IR (Higado)

(iii) Mismatch of searcher's vocab vs. collection vocab (cosmonaut / astronaut)

A(2) There are several relevance prototypes. → (true doc ki representation bhaut hai & we can move our query there)

eg. (i) Burma / Myanmar (ii) pop stars that worked at Burger King.

→ Often: instances of a general concept ; good editorial content can address prob.

9.1.4 – Relevance Feedback on Web

Some web search engines offer a similar/related pgs. feature – the user indicates a doc in the results set as examplary from standpoint of meeting his info need & requests more doc like it.

Eg. Sprinkle et. al (2000) presents result using excite Search Engine.

9.1.5 – Evaluation of relevance feedback strategies

① Interative Rf can give very substantial gains in ~~search~~ retrieval performances.

→ One round of rf is v useful, second round (2) are more useful. Atleast five judged doc is recommended

② Use doc in residual collection (set of doc minus those assessed relevant) for 2nd round of evaluation

↑ neither of the two is fully satisfactory

∴ best is do a time based comparision ie. how fast does a user find Rel doc with rf vs another strategy (eg. query reformation)

Day / Date: _____

9.1.6 — Pseudo Relevance Feedback

→ Aka blind rf provides a method for ~~auto~~ automatic local analysis.

→ Automates the manual part of rel f so the user gets improved result without an extended interaction

• Method → do a normal retrieval to find an initial set of top K ranked doc are relevant &
do relevance fd as before on this assumption.

9.1.7 — Indirect Relevance feedback (aka implicit rf)

Relevance feedback Problems

① long queries are inefficient for typical IR engine

② long time taken for response by users

③ Higher cost for retrieval sys ~~④ Partial sol.~~

④ Users are often reluctant to provide explicit feedback.

⑤ Often hard to understand why a particular doc was retrieved after applying rf

9.2 — Global Methods for Query Reformulation

Expanding query by simply ~~adding~~ aiding the user in doing so, by using Thesaurus.

9.2.1 — Vocabulary Tools for Query Reformulation

IR sys ~~must~~ might suggest search terms by means of a thesaurus or a controlled vocab.

9.2.2 — Query Expansion

→ User gives additional input (good/bad search term) on query words or phrases, possibly suggesting additional query terms

→ Query expansion help in extending user queries with related terms in order to solve the lexical gap prob in IR

→ Most common form of query expansion is global analysis using some thesaurus.

*Use of Thesaurus can be combined with ideas of term weighting.

Day / Date: _____

- How to augment user query?

→ Methods for building a thesaurus for query expansion includes:

① Use of controlled vocab is maintained by human editors (can be query rather than just synonyms)

② An automatic derived thesaurus

③ Query Reformulations based on query log mining.

→ Thesaurus based query expansion has the advantage of not requiring any input.

→ Query exp. increases recall → effective ; may significantly ↓ precision.

→ Global analysis (static of all docs in C)

- Automatically derived thesaurus → generate a thesaurus automatically by analyzing a collection of docs. There are two ways:

① * Exploit word coocurrence (corpus text statistics to find most similar words

② * Use shallow grammatical analysis of text and to exploit grammatical relations

In a nutshell,

Query Expansion is less successful than relevance feedback; however, it is more understandable to sys user.

Dazzle

Day / Date: _____ Note: This chp is till 11.3

(Chp 11) # PROBABLISTIC INFORMATION RETRIEVAL

11.1 – REVIEW OF BASIC PROBABILITY THEORY

→ Joint probability = $P(A, B)$

→ Conditional probability = $P(A|B)$

} → Relation b/w both & Chain Rule:

$P(A, B) = P(A \cap B) = P(A|B)P(B) = P(B|A)P(A)$

→ Partition Rule ⇒ $P(B) = P(A, B) + P(\bar{A}, B)$

* $P(A_1) = P(A_1, B_1) + P(A_1, B_2)$
* $P(A_2) = P(A_2, B_1) + P(A_2, B_2)$

→ Bayes's Rule ⇒ $P(A|B) = \frac{P(B|A)\,P(A)}{P(B)} = \left[\frac{P(B|A)}{\sum_{x \in \{A, \bar{A}\}} P(B|x)\,P(x)}\right] \cdot P(A)$

↳ this eq. can be thought of as a way of updating probabilities.

(i) Start off with prior prob. $P(A)$ {initial estimate of how likely event A is in absence of any other info}

(ii) Derive a posterior prob $P(A|B)$ after having seen the evidence B, based on the likelihood of B occuring in 2 cases that A does/doesn't hold.

→ Odds of an event, a kind of multiplier for how prob changes:

Odds – $O(A) = \frac{P(A)}{P(\bar{A})} = \frac{P(A)}{1 - P(A)}$ ← { $P(A) \approx \text{Odd}(A)$

↳ success : failure (ratio)

Why probability in IR?

① → IR deals with uncertain info ; Prob → measure of uncertainity.

② → Probabilistic Ranking Principle → • Provable • Minimization of risk.

③ → Probabilistic ~~ref~~ inference → To justify your decision

Basic probability

- Probability is study of randomness & uncertainity.
- favourable events / total event space in consideration.
- no. assigned to each event (subset) of a sample space Ω.

Dazzle

① * if $A \in \Omega \rightarrow P(A) \geq 0$

② * $A, B \in \Omega$ $A \subseteq B \rightarrow P(A) \leq P(B) \geq 0$

Note: E & E' are mutually exclusive so $P(E) = 1 - P(E')$

Day / Date: _____

↑

Axioms of Probability

→ for any event A : $0 \leq P(A) \leq 1$

→ $P(\Omega) = 1$ → (entire sample space)

→ if $A = A_1 \ldots A_n$, then $P(A) = P(A_1) + \ldots P(A_n)$

→ $P(\emptyset) = 0$

Properties of Probability

→ $P(A^c) = 1 - P(A)$

→ if $A \subset B$, then $P(A) \leq P(B)$

→ $P(A \cup B) = P(A) + P(B) - P(A \cap B)$

→ $P(A \cup B \cup C) = P(A) + P(B) + P(C) - P(A \cap B) - P(A \cap C) - P(B \cap C) + P(A \cap B \cap C)$

→ Independence ⇒ $P(A, B, C) = P(A) + P(B) + P(C)$; all are independent events.

11.2 – THE PROBABILITY RANKING PRINCIPLE

11.2.1 – 1/0 loss case

We assume that for a query q and doc. d in collection, value = 1 if Relevant else val = 0.

imp

→ In prob. model, docs are ranked in decreasing order of their probability with prob. estimated accurately will into need : $P(R=1 \mid d, q)$. The overall effectiveness of the sys to its user will be the best that is obtainable on the basis of those data.

→ In simplest case of PRP, there is no retrieval costs or other utility that would differentially weight actions or errors. You lose a point for either returning a NRel doc or failing to return a doc. The goal is to return best possible top results as top k doc, for any val. of k (user's choice)

^ Such binary situation where you are evaluated on your accuracy is called 1/0 loss.

If set of retrieved doc is to be returned rather than ordering, then Bayes Optimal Decision Rule is to be applied. The decision which minimizes the risk of loss → return doc than relevant (more likely)
(under 0/1 loss)

d is relevant iff $P(R=1 \mid d, q) > P(R=0 \mid d, q)$

$P(R|x) = P(x|R) \cdot P(R) / P(x)$; $P(NR|x) = P(x|NR) \cdot P(NR) / P(x)$ → if $P(R|x) > P(NR|x)$ → then x is relevant

11.2.2 – The PRP with retrieval costs

→ Assume model of retrieval cost → C_0 = cost of not retrieving a rel doc → C_1 = cost of retrieving a non-rel doc.

for doc 'd' & for all docs d' → not yet retrieved

$$C_0 \cdot P(R=0 \mid d) - C_1 \cdot P(R=1 \mid d) \leq C_0 \cdot P(R=0 \mid d') - C_1 \cdot P(R=1 \mid d')$$

Dazzle

Applications of PRP:
- Find prob. of relevance with a fixed collection with fixed set of queries.
- Find ranking of doc for a given query, when relevance is given for a set of docs.
- " " " " " " " " is not given.

Day / Date:

then d is the next doc to be retrieved. Such model gives a formal framework where we can model differential costs of false +ve & false -ve & even sys performance at modelling stage.

11.3 — BINARY INDEPENDENCE MODEL

→ BMI has been traditionally used in PRP.

→ Assume: Binary = boolean ; doc & queries both represented as binary term incidence vectors.

* doc d is represented by vector $\vec{x} = (x_1, \ldots x_m)$, where $x_t = 1$ if term t present in doc else $x_t = 0$

* q is represented by $\vec{q}$.

* Independence means that terms are modelled as occuring in doc independently. No association b/w terms.

* The representation is an ordered set of Boolean variable.
* Many docs can have same vector representation with this simplification.

~~→ We assume that relevance of each doc is independent of relevance of other doc~~

→ Under BMI, we model the prob $P(R|d,q)$ that a doc is relevant via prob in terms of term incidence vectors $P(R|\vec{x},\vec{q})$. Then using Baye's Rule:

$$P(R=1|\vec{x},\vec{q}) = \frac{P(\vec{x}|R=1,\vec{q}) \cdot P(R=1|\vec{q})}{P(\vec{x}|\vec{q})}$$

$$P(R=0|\vec{x},\vec{q}) = \frac{P(\vec{x}|R=0,\vec{q})\, P(R=0|\vec{q})}{P(\vec{x}|\vec{q})}$$

→ if rel or nrel is retrieved then representation of doc is $\vec{x}$

also, $P(R=1|\vec{x},\vec{q}) + P(R=0|\vec{x},\vec{q}) = 1$

x —— x continued after deriving PRP

~~11.3.1 — Deriving a ranking function for query terms~~

~~We wish to order returned doc by descending → Under BMI, it is modeled as $P(R=1|\vec{x},\vec{q})$.~~

~~$$O(R|\vec{x},\vec{q}) = \frac{P(R=1|\vec{x},\vec{q})}{P(R=0|\vec{x},\vec{q})} = \frac{P(R=1|\vec{q}) \cdot P(\vec{x}|R=1,\vec{q}) / P(\vec{x}|\vec{q})}{P(R=0|\vec{q}) \cdot P(\vec{x}|R=0,\vec{q}) / P(\vec{x}|\vec{q})}$$~~

~~] relevance odd] non rel. odd (expansion of baye's rule)~~

~~rank decreasing ↓ (odd mein transform kiya hai)~~

~~$$= \frac{P(R=1|\vec{q})}{P(R=0|\vec{q})} \cdot \frac{P(\vec{x}|R=1,\vec{q})}{P(\vec{x}|R=0,\vec{q})}$$~~

Dazzle

Day / Date: _____

→ Deriving PRP ranking principle.

~~11.3.1 — Deriving a ranking function for query terms~~

— Assumptions that we'll follow:

A1 — are random variable for each term (word) ; A2 — $d(w)$ are mutually independent given R.

A3 — $P(0/R=1) = P(0/R=0) = 0$

A4 — If word is not in query, it is equally likely to occur in R & NR doc; only need to calculate prob of common words in query & docs.

A5 — On avg, a query word will occur in half of the rel docs; $P(w)$ & $1-p(w)$ will cancel out.

A6 — NR set approximated by collection as a whole (most docs are NR).

— Derivation of PRP

→ Assume, $D = \{d_1, d_2, \ldots d_n\}$ & q is a fixed query. ; $|D| = n$

→ As per PRP, for a given q, we can rank doc as decreasing order of probability of relevance.

$\forall_{i=1}^{n}$ rank d according $P[R=1/d,q]$ — formula of ranking (compare every doc with q to see if rel. or not)

Representing in vector → $P[R=1/\vec{x},\vec{q}] = P[R=1/\langle t_1 \ldots t_n \rangle, \vec{q}]$

→ This ranking of doc can be represented in prob. of odd: ($\vec{x}$ = doc)

$$O[R/\vec{x},\vec{q}] = \frac{P[R=1/\vec{x},\vec{q}]}{P[R=0/\vec{x},\vec{q}]}$$

$O[R/\vec{x},\vec{q}]$ ↓ odd prob. — numerator: +ve event; denominator: reciprocal event

now expanding using baye's rule:

$$\frac{\text{relevance odd}}{\text{NR odd}} \quad \frac{(P[R=1/\vec{q}] \cdot P[\vec{x}/R=1,q])/P[\vec{x}/\vec{q}]}{(P[R=0/\vec{q}] \cdot P[\vec{x}/R=0,q])/P(\vec{x}/\vec{q})}$$

Here, $P(\vec{x}/\vec{q})$ will cancel out whereas, $P[R=1/\vec{q}]$ & $P[R=0/\vec{q}]$ are constant as per assum (A4)

$$\therefore O[R/\vec{x},\vec{q}] = \frac{P[\vec{x}/R=1,\vec{q}]}{P[\vec{x}/R=0,\vec{q}]}$$

writing in vector form:

$$\prod_{t=1}^{n} \frac{P[\vec{x}_t/R=1,\vec{q}]}{P[\vec{x}_t/R=0,\vec{q}]} \cdot O(R/\vec{q})$$

n: total no. of terms; $O(R/\vec{q})$: fixed (can be removed)

→ for each term that occur in rel doc for query " " " " " " " NRel doc " "

Day / Date: _____

Let, $P_t = P(x_t = 1 / R = 1, \vec{q})$; P_t → (count of) kitnay rel doc mein term has appeared / total relevant

$u_t = P(x_t = 1 / R = 0, \vec{q})$; u_t → kitnay non-rel doc mein term has appeared.

We'll get the ~~confusion~~ contingency matrix as:

	$R=1$	$R=0$
term appeared in doc ← $x_t = 1$	P_t	u_t
(didn't) ← $x_t = 0$	$1-P_t$	$1-u_t$

Now, we have:

$$O[R|\vec{x}, \vec{q}] = \prod_{t=x_t=q_t=1} \frac{P_t}{u_t} \cdot \prod_{x_t=0\ q_t=1} \frac{1-P_t}{1-u_t}$$

term appeared in doc & query both — term not in doc but in query

Now multiply above eq. by:

$$\prod_{t=x_t=0\ q_t=1} \frac{1-P_t}{1-u_t}$$ is a constant

$$O[R/\vec{x}, \vec{q}] = \prod_{t=x_t=q_t=1} \frac{P_t(1-u_t)}{u_t(1-P_t)} \cdot \prod_{t=q_t=1} \frac{1-P_t}{1-u_t}$$ cancels out b/c of AS

So, this odd $[R=1/\vec{x}, \vec{q}]$ can be used as a Retrieval status value.

$$RSV = \prod_{\underbrace{t=x_t=q_t=1}} \frac{P_t(1-u_t)}{u_t(1-P_t)}$$

jo terms common hain unke → is ratio ko (*) multiply kardein

& finally,

$$\prod_{t=x_t=q_t=1} \frac{P_t/(1-P_t)}{u_t/(1-u_t)}$$

this formula will be used for calculation & implementation

→ we can use logarithm to transfer this into summation:

(see video of April '02 → 30 mins onward for application (numerical qs))

Dazzle

Day / Date: _____

→ Estimating P_t & U_t

→ let N be no. of docs in corpus ; → R be #rel doc in sample.

→ n_t → no. of docs that contains term 't' ; → r_t → # rel doc that contains t.

$$P_t = \frac{r_t}{R} \qquad U_t = \frac{n_t - r_t}{N - R}$$

total no. of rel doc ← R

for NR doc N − R → total no. of NR doc

Lindstone Smoothing → ($\lambda = 0.5$) → to avoid val = 0.

$$P_t = \frac{r_t + 0.5}{R + 1} \qquad U_t = \frac{n_t - r_t + 0.5}{N - R + 1}$$

→ opar + 0.5
→ neechay + 1

→ (introduced in general term)

BINARY INDEPENDENCE MODEL

● we get the below eqn. after doing basic steps of PRP:

$$O(R|\vec{x},\vec{q}) = O(R|\vec{q}) \cdot \prod_{t=1}^{m} \frac{P(x_t|R=1,\vec{q})}{P(x_t|R=0,\vec{q})}$$

divide the terms to give:

$$O(R|\vec{x},\vec{q}) = O(R|\vec{q}) \cdot \prod_{t:x_t=1} \frac{P(x_t=1|R=1,\vec{q})}{P(x_t=1|R=0,\vec{q})} \cdot \prod_{t:x_t=0} \frac{P(x_t=0|R=1,\vec{q})}{P(x_t=0|R=0,\vec{q})}$$

(U_t) term in NR rel doc ← term in rel doc (P_t)

term appearing in doc — term not appearing in doc

we get contingency table:

doc	relevant (R=1)	non-rel (R=0)
Term present	$x_t = 1$	P_t	U_t
Term absent	$x_t = 0$	$1 - P_t$	$1 - U_t$

→ constant for the particular query

$$\therefore O(R|\vec{q},\vec{x}) = \boxed{O(R|\vec{q})} \cdot \prod_{t:x_t=q_t=1} \boxed{P_t/U_t} \cdot \prod_{t:x_t=0,q_t=1} 1-P_t/1-U_t$$

query term found in doc — query term not found in doc

Dazzle

Day / Date: _____

$h(i,j)$ = term appearing in doc (rel.) / non-rel.

so, $RSV_d = \log \left[\prod_{t: x_t = q_t = 1} \frac{p_t(1-u_t)}{u_t(1-p_t)} \right] \Rightarrow \boxed{c_t = \log \frac{p_t}{1-p_t} + \log \frac{1-u_t}{u_t}}$

where, c_t → log odd ratios for term in q.

→ $c_t = 0$ if t has equal odds of appearing in rel & NRel docs

→ $c_t = +ve$ if t has more odds of appearing in a rel doc.

Probability estimates in theory

	doc	rel	non-rel	total
Term present	$x_t = 1$	s	$df_t - s$	df_t
Term absent	$x_t = 0$	$S - s$	$(N - df_t) - (S - s)$	$N - df_t$
	Total	S	N − S	N

$f(x) = f(u) + f(v)$ (extra)

df_t → no. of doc that contains t

S → relevant

$df_t - s$ → non relevant

N → total no. of doc

$N - df_t$ → doc in which t doesn't appear

$S - s$ → rel. doc but t doesn't appear

Using this,

$p_t = s/S$ & $u_t = (df_t - s)/(N - S)$

RSV_d → $\boxed{c_t = K(N, df_t, S, s) = \log \left| \frac{s/S - s}{(df_t - s)/((N - df_t) - (S - s))} \right|}$

→ RELEVANCE FEEDBACK IN PROB IR

① Guess initial estimates of p_t & u_t. eg. assume p_t is constant all over x_t in q so $p_t = 1/2$

② Use current estimate to guess best set of rel doc → $R = \{d : R_{d,q} = 1\}$; present this rel doc to user.

③ Interact with user to refine model of R → we do this by learning from user rel. judgement for subset of doc V.

V is partitioned in 2 subsets → $VR = \{d \in V, R_{d,q} = 1\} \subset R$ & $VNR = \{d \in V, R_{d,q} = 0\}$ → disjoint from R.

④ Re-estimate p_t & u_t on basis of known Rel & NRel doc. If sets VR & VNR are large enough, we may be able to estimate quantity directly from those docs → $p_t = |VR_t|/|VR|$; where VR_t → set of docs in VR containing t.

$$p_t = \frac{|VR_t| + 1/2}{|VR| + 1}$$ → general practice of doing smoothing.

Dazzle

Day / Date: _____

→ however, set of docs judged by user (V) is v. small so resulting stats is unreliable ∴ we combine new info with original guess in a process of Bayesian update : $p_t^{(k+1)} = \frac{|VR|\, r_t + k p_t^{(k)}}{|VR| + k}$

Here, $p_t^{(k)}$ is k^{th} estimate for p_t in an iterative updating process & is used as Bayesian prior in next iteration with the weighting of k.

→ Prior is strongly weighted so that the estimate doesn't change too much from the evidence provided by a very small no. of docs.

⑤ Repeat above process from step 2, generating a succession of approximations to R & hence p_t, until user is satisfied.

RELEVANCE FEEDBACK IN PROB IR

① Assume initial estimates for p_t & u_t as above.

② Determine a guess for size of rel doc set. If unsure, a conservative (too small size) guess is likely to be best. This motivates use of a fixed size set V of highest ranked doc.

③ Improve guess for p_t & u_t. Re-estimate p_t based on set V instead of VR. let V_t be subset of doc in V.

$$p_t = \frac{|V_t| + \frac{1}{2}}{|V| + 1}$$ and if we assume that docs that are Non retrieved are non rel.

then we can update our u_t estimate as: $u_t = \frac{df_t - |V_t| + \frac{1}{2}}{N - |V| + 1}$

④ Go to step 2 until ranking of returned results converges.

⇒ Once we have real estimated val. for p_t, then c_t used in RSV val. looks almost like tf-idf value.

ISSUES with PROBABILISTIC IR

→ Term independence (assuming all terms are independent which is not true. Some terms might appear in doc b/c of others) (joint prob not calculated)

→ Terms not in query don't affect the outcome (not true; synonyms or same concept may affect)

→ Boolean representation of docs/queries/relevance (Binary representation only; no partial)

→ Doc relevance values are independent (not true, because docs might be inter-related & relevancy might change in relation to other)

Dazzle

Day / Date: _____ { GENETIC ALGO EXAMPLE }

Example of PP

TEXT CLASSIFICATION & NAIVE BAYES (Chp 13)

$D = \{d_1, d_2 \ldots d_n\}$

$C = \{c_1, c_2\}$ → no. of classes are fixed ; given

$\forall_{d} \rightarrow c_j$ { for all docs classify to which class do they belong

eg. classify mails if spam/ham

Ad Hoc Query — Free text query — search engine supports it (any random query need of user)

Standing Query — Executed on a collection of changing doc eg. new cases of corona virus.
(ran multiple times)

Text/Doc Classification

Text → can be anything ; sms, email, book, researches (no boundary). (general topics)

Doc → has a particular genre.

(unit text ka aik doc, doc → one tweet, or a research paper)

Text/doc classification is process of assigning or a predefined set of classes.

* ~~we refer to this~~ classification using standing query is also called routing/filtering.

TYPES OF CLASSIFICATION

→ ① Manual

Manually mark (how you block a specific no. or email id)

It is challenging b/c we manually cater.

Manual Hand Code Rules → Rules based on combination of words or other features.

→ ② Supervised

A very large dataset available. We know every doc class & all is easily available.

$\vec{d_1} = \vec{x_1}$; $\vec{x_1} = \langle t_1, \ldots t_m \rangle$; $\vec{x} \rightarrow \hat{y}$ (doc → class)

Dazzle

Day / Date: SE (Clip4) (not in book)

© USER STORY

① Training req → ② Testing → test krtay hain ispe how efficient our result is

f(d) → C; who + what + why → imp

* Labelled data is available (rich) & we build classifier by making our program learn through this data. product owner or customer start writing cards which is a high

→ A training set of m hand labelled dataset

(5 W ques ans)

→ ③ Semi supervised ion

→ Start as supervised. ss analyst & developer together ask the customer

→ Tune in such a way that it will continue to learn from functionality of its working

<classification> on

* jo labelled dataset hai us se new doc ko classify kreinge & check that data & train further.

→ Form of func. in card

→ ④ Unsupervised type I want to) I want to perform an action so that

Training dataset has labels but we give only doc to classifier & check if it has accurately classified.

→ (learn on its own how to classify text/doc by implicitly learning the features)

13.2 — NAIVE BAYES TEXT CLASSIFICATION (NB)

→ first (baseline) & basic supervised learning method — Multinomial Naive Bayes — Probabilistic learning method

→ The probability of a doc d being in class c is:

$$P(c|d) \propto P(c) \prod_{1 \le k \le n_d} P(t_k|c)$$

where,

$P(t_k|c)$ → prob of term t_k occurring in doc of class c. → measure

$P(c)$ → prior probability of doc occurring in class c

n_d → no. of tokens in d

Dazzle

→ simple classifier based/works on prior prob. of classes' relationship.

Day / Date: _____

In text classification, our goal is to find best class for doc. Best class in NB is mostly likely or maximum posteriori (MAP) class c_{map}:

$$c_{map} = \arg\max_{c \in C} \hat{P}(c|d) = \arg\max_{c \in C} \hat{P}(c) \prod_{1 \le k \le n_d} \hat{P}(t_k|c)$$

$\hat{P}$ → represent P b/c we dk true values of parameter $P(c)$ & $P(t_k|c)$, but estimate them from training set.

for argmax, we'll calculate for everyone. all c belongs to class (c). So max hoga that will be the class most likely.

$$c_{MAP} = \arg\max_{c \in C} P(c|d)$$

$$= \arg\max_{c \in C} \frac{P(d|c)P(c)}{P(d)}$$ → applying Baye's Rule on

$P(d)$ → constant → doesn't affect ranking.

$$= \arg\max_{c \in C} P(d|c)P(c)$$ ← so we'll drop the denominator

→ HOW TO CALCULATE THESE PROBABILITIES?

To first attempt → max likelihood estimates → simply use freq in data (counting technique)

① $\hat{P}(c_j) = \frac{doc\_count(C=c_j)}{N_{doc}}$ → for a given class j ← P for a given class in training dataset

(so we'll calculate prob. of every class) doc_count → for given class in training dataset; N_{doc} → total no. of doc in training dataset.

② $P(w_i|c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}$ ← prob. of a word i belong to class j

→ count of word i in class j

→ total count of word i in all classes

(set of words; V → set of vocab)

① → formula for classes

② → formula for words → after smoothing → $\hat{P}(w_i|c) = \frac{count(w_i, c) + 1}{\sum_{w \in V} (count(w, c)) + 1}$

$$= \frac{count(w_i, c) + 1}{\left[\sum_{w \in V} count(w, c)\right] + |V|}$$

|V| → count of vocab.

Dazzle

min client download time.

time = F/Dmin

time to distribute F to N client using c-s approach $D_{c-s} \geq \max(NF/u_s, F/d_{min})$ ↑(total time)

File dist. time: P2P

Day / Date:

everyone is downloading & jo down. kr raha wo upload bhi kr raha.

server trans only 1 copy → F/us ; each client must download → F/dmin

client as Aggregate must download NF bit = max upload rate → $u_s + \Sigma u_s$ limiting max d. rate

(total down. data)

→ time to dis. F to N client using P2P $\max(F/u_s, F/d_{min}, NF/(u_s+\Sigma u))$

→ DNS (Domain Name Sys) (32 bit - 4 octats)

host name → convert kr ke IP address ko deta hai

use ↑ ds time ↑

① • distributed database → implemented in hierarchy of many name server.

② • App-layer protocol → host, name server communicate to resolve name (address & name data)

• DNS: services + structure

① → hostname to IP address translation karta hai.

② → host aliasing : conical name

↓

host with a complicated host name can have one or more alias name.

③ Mail server aliasing

④ load distribution – for replicated web server, a set of IP address is thus associated with one canonical hostname. (many IP add. corresponds to one name) ^

→ DNS rotation distributes the traffic among the replicated servers.

→ also used for email server.

Why not centralize DNS?

→ single point of failure → traffic volume → maintenance →

DNS: root name server

→ contacted by local name server that cannot resolve name.

→ root name server → contact authoritative servers. → top level Domain (TLD) servers (root TLD to query forward kr ke ga)) → authoritative DNS server.

↓ auth. to IP address

* local DNS server (name.)

→ has local cache of recent name to add. translation ; acts as a proxy → forward query to hierarchy.

(will search authoritative ki ko represent kr raha ho)

* DNS name resol.

↳ host at loc. poly

① → iterative query

② → recursive query

→ DNS caching, updating record

• once (any) name server learns mapping, it caches mapping.

→ cache entries timeout (disappear) after some time (TTL)

• cache updating

• DNS records (has types) → Distributed ~~dats~~ db storing resource records (RR) → RR format: (name, val, type, ttl)

• DNS protocol → query & reply msg. both have same msg format. → 16 bit # & flags

• Inserting record in DNS → reg name at DNS registrar ; provide name, IP add of author. name server)

→ create authoritative server type A for records for networkutopia.com.

pure P2P architecture : no always-on server ; arbitrary end sys directly comm. ; peer are intermittently

File distribution : client server vs. P2P → ask sys pc dependability nahi hoti — connected peer 2 peer

n server min to n copies.

time to send 1 file → F/us → speed. ← file size

NF/u_s → for n files

Day / Date:

website use cookie 4 components.

User-server state : cookie → (saves user state)

↳ server apni header file mein client ko id dega

↳ server se sari comm. us id pe against hogi — history maintain.

(database browser aur server dono sides pe ho raha hai)

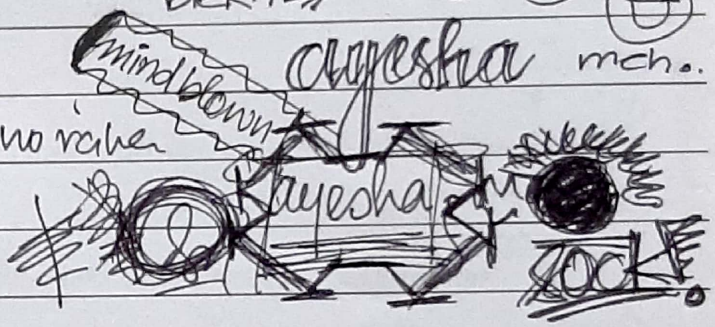
for each web a new id is given to user.)

↓

web transaction ki state maintain no rahe

HTTP → stateless

↳ but HTTP msg carries state.



Web cache (proxy server) → get method se web pg search krein

GET → if normal web page on browser

Post → if send a form to server.

→ if first time, no copy on proxy → it will send pg

→ if copy available in cache, cache send obj

↓ as a client when copy not available

↓ as a server when copy available

* cache act both as server & client

(web cache reduces response time)

→ (cache example read from book)

→ (RTT → bohot zyada hojata hai)

(2 type of delay)

↓

web cache → speed high & cheaper.

CONDITIONAL GET → Proxy server ko bta rahay agr page ki copy hai

tou req. page send na krein server to page.

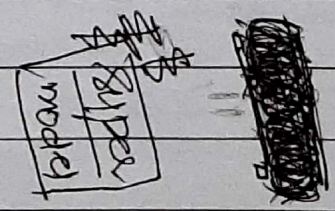
Electronic Mail

→ user agent (email generate)

→ mail server (saving email)

→ simple mail transfer protocol: (will communicate with main server)

SMTP → (uses TCP to reliably send)



3 main components

mail access protocol

POP3 + IMAP

Day / Date: _____

$$P(c_j|d) = P(d|c_j) \cdot P(c_j)$$

MULTINOMIAL NB (Algorithm)

→ Training → $D = \{d_1, d_2 \ldots d_n\}$; $C = \{c_1, c_2 \ldots c_n\}$ → given

Train(C, D) {

① Extract vocab (clean data) from D as V.
V ← ExtractVocab(D)

② Count no. of docs as N. → $n = |D|$
N ← countdocs(D)

③ for each class, calculate posterior prob from prior prob.
for each $c \in C$

④ Take out ~~do~~ count of doc belonging to that class as Nc.
do Nc ← countDocInClass(D, c)

⑤ prior prob → Nc/N. ← $= P(c_j)$
{ prior[c] ← Nc/N

⑥ $text_c$ mein combine all the text of those doc that belong to c_j
$text_c$ ← concatenate(D, c)

⑦ then for each term that belongs to Vocab.
for each $t \in V$

⑧ calculate count of term that belongs to $text_c$ as T_{ct}
do T_{ct} ← countterm ($text_c$, t)

⑨ & then for each term that belongs to Vocab.
for each $t \in V$

⑩ calculate conditionalprob[t][c] i.e. $\frac{T_{ct}+1}{\sum_t (T_{ct}+1)}$ → $= P(d|c_j)$
do condprob[t][c] ← $\frac{T_{ct}+1}{\sum_t (T_{ct}+1)}$

⑪ Returns V (vocab), prior prob & ~~conditional~~ prob for all classes — return V, prior, condpr

→ Testing → C, V, prior, condprob, & new doc d given (decide to which class d belongs)

Apply NB(C, V, prior, condprob, d) {

① Extract words from the doc using vocab as W.
W ← ExtractTokensFromDoc(V, d)

② for each class i that belongs to C
for each $c \in C$

③ calculate prior prob of every class as score
do score[c] ← log prior[c]

④ for each term (t) that belongs to word (W)
for each $t \in W$

⑤ add ~~calculate~~ condprob in score & save it in score
do score[c] += log condprob[t][c]

⑥ return the max score for showing to which class d belongs to.
return $\underset{c \in C}{\text{argmax}}$ score[c]

for example goes (numerical → see april 9 video .35 min onwards)

→ agar testing mein koi aisi term ajaye which doesn't belong to training data, we calculate its prob. by $P\left(\frac{1}{V+1}\right)$ → add 1 in vocab eg. neha. & |vocab| = 6 so $P(\text{neha}) = \frac{1}{41} = \frac{1}{7}$

Dazzle

Day / Date: _____

(computationally lengthy)
doc ke token se class ka andaza

VARIATIONS OF NAIVE BAYESIAN

→ Two ways in which we can setup NB classifier : ① Multinomial NB ② Multivariate Bernoulli model

→ MULTIVARIATE BERNOULLI MODEL

→ Works on doc count not on features.

Training: TrainBernoulli(C, D)

```
1. { V ← ExtractVocab(D)
2.   N ← count_docs(D)
3.   for each c ∈ C
4.   do Nc ← countDocInClass(D, c)
5.      prior[c] ← Nc/N
6.      for each t ∈ V
7.      do Nct ← countDocInClassContainingTerm(D, c, t)
8.         condprob[t][c] ← (Nct + 1)/(Nc + 2)   (smoothing)
9.   return V, prior, condprob
```

Testing: ApplyBernoulli(C, V, prior, condprob, d) {

```
1. Vd ← ExtractTermsFromDoc(V, d)
2. for each c ∈ C
3. do score[c] ← log prior[c]
4.    for each t ∈ V
5.    do if t ∈ Vd
6.       then score[c] += log condprob[t][c]
7.       else score[c] += log(1 − condprob[t][c])
8. return argmax_{c∈C} score[c]
```

→ For example numerical see WII slide no. 11, and for differences b/w both types too.

Dazzle

(*accuracy increases b/c of sidenarys allowed)

Day / Date: _____

EVALUATION OF CLASSIFICATION TASK

	spam	ham
Predicted spam	35	16
Predicted ham	15	54

Accuracy = 35+54/120

Error = 15+16/120

Recall — fraction of docs in class i classified correctly (kitna accurately class ko identify kr rahay)

→ rel retrieved / total relevant ; $c_{ij} / \sum_j c_{ij}$

Precision — Fraction of docs assigned class i that are actually about class i. → rel. / total retrieved.

$= c_{ii} / \sum_j c_{ji}$ → 0+0+0+0 → sum.

→ sum. (diagonals ka sum col. ki)

Accuracy — Fraction of docs classified correctly ie(1 − error rate).

→ For calculating combined multiple performance measure, we use micro / macro avg

- **Macroaveraging** — compute performance for each class, then avg.
- **Microaveraging** — collect decisions for all classes, compute contingency table, evaluate.

eg:

c_1	yes	no
yes	10	10
no	10	970

c_2	yes	no
yes	90	10
no	10	890

microavg table

	yes	No
yes	100	20
No	20	1860

→ Macroavg

~~Microavg~~ P = (0.5+0.9)/2 = 0.7

↓ (calculate seperately)

→ Microavg P = 100/120 = 0.83 $\frac{(c_1+c_2)}{2}$; microavg score is dominated by score on common classes

SUPERVISED LEARNING — DATASETS

eg. X → Y ; for binary classification eg. Spam/Ham

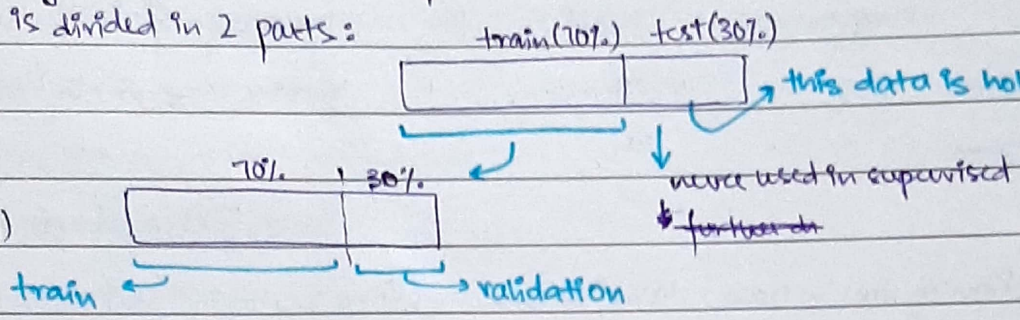
$e_1 = f_1 \ldots f_m$ — Y

↓ features ↓ classes

Day / Date: _____

→ Model fitting → feature selection / parameter estimates.
labelled

- Dataset is divided in 2 parts:



↑ this approach is called HOLDOUT aka split approach.

→ 30% validation is used for testing whereas rest is used for training.

→ for cross validation & better model performance, we use k-fold technique.

k → no. of parts & use k-1 parts to train model on k different parts of data & use 1 part for testing & avg out the performance of all models.

Variance vs. Bias → two types of Errors

error due to:

Biasness → Difference b/w expected & actual

Variance → variability of model.

eg: bulls eye. → If low var. & low biasness so near target performance.

CLASSIFICATION MODEL & ACCURACY

→ classification model learning is a challenging task.

- Textual docs are rich in feature → take word to its relevant space. (make a group) cluster near words.
- Feature selection is helpful in simplifying the model & improving accuracy of model.

→ Textual features: Words/Tokens, Phrases/Bigram, Graph, NLP based features.

→ **FEATURES**

An individual measurable property of a phenomenon being observed.

→ Choosing informative, discriminating & independent features is a crucial step for effective algo in pattern recognition, classification & regression.

Day / Date: _____

→ Features are usually numeric, but structural feature such as strings & graphs are used in syntactic pattern recognition.

FEATURE SELECTION

→ Purpose of selection of subset of features for training set & using only this subset as features in text classification.

→ Feature selection serves two main purposes:

① It makes training & applying a classifier more efficient by decreasing the size of effective vocab.

② Feature selection often increases classification accuracy as noise features are eliminated through it.

→ Types of Selection Feature Method:

① WRAPPERS

→ Jo features nikalay hain us se dekhein ge ke wo target ko kitna behtar predict kr rahay. We make a classifier using these features & then measure the accuracy of the features. & select the features that are highly accurate.

→ Wrappers use classification accuracy of some learning algo as their evaluation function.

→ When no. of features is high, they are time consuming b/c wrappers train a classifier for evaluating each subset feature.

→ Wrappers not suitable for text classification.

② FILTERS

→ Performs feature selection independently of learning algo that will use selected features. (first randomly filters features & then check accuracy)

→ For evaluation of a feature, filter use an evaluation metric that measures the ability of feature to differentiate each class.

→ Much less time consuming than wrappers & widely used in text classifi.

Day / Date: _____

③ EMBEDDING – (new technique)

→ Use a group of features. We take projection of a particular word from relevant space & place the group of features at that place.

(Ek word ko uski relevant space se utha ke usse as pas ke words ka group bana ke us word ki jagah place kr dete hain). Different target space mein different features.

- **Forward Feature Selection (FFS)** — Ek feature select kr uski accuracy, phr 2, then 3 and so on. Make numerous combinations.
- **Backward Feature Selection** — n no. of features ki accuracy, then n-1, then n-2 & so on.

HOW FEATURE SELECTION WORKS?

→ FS replaces a complex classifier (using all features) with a simpler one (using subset of features).

→ Purpose of feature selection algo is to select only those features that ~~to~~ compute utility measure A(t,c) for each term of vocab & select the k terms that have highest val. of A(t,c).

* A(t,c) → function A to calculate val. of feature t & in class c to see if t should be added in classifier or not.

→ Algo of feature selection in (fig-13.6)

Underfitting → classifier has less knowledge due to which it classifies wrong.

Overfitting → Model is trying to adopt all features & makes function complex (many parameters) (none of these are desirable) * we require best fitting. Selection of relevant features

→ size, colour, shape & material are imp features.

Feature Extraction ~ Similar to feature selection but we extract relevant feature from an image that is relevant to the domain & train model with that sufficient knowledge.

Dazzle

Day / Date: _____

THREE APPROACHES TO FEATURE SELECTION

① Mutual Info (MI) , $A(t,c) = I(U_t ; C_c)$

② Chi square , X^2 , $A(t,c) = X^2(t,c)$; general lib → chi2 , X^2 , xsq()

③ Freq based features , $A(t,c) = N(t,c)$ ie (tf*idf)

Noise removal / Outliers → Values that are outside our decision boundary

↳ challenge in Text

• MUTUAL INFORMATION

(feature class se kitna related hai aur kitna help kr rahi hai class identify krnay mein)

→ MI measures how much info the presence/absence of a term condition contributes to making the correct classification decision on c.

$I(U;C)$ — random var. for term ↵ " " classes ↵

— N_{11} : term present & belongs to class
N_{10} : term " but doesn't belong to class
N_{00} & N_{01} .

→ Example 13.3 for numerical.

• CHI SQUARE METHOD

→ Important for text classification.

Null hypothesis : H_0 → (t,c) → feature 't' & class 'c' are not significantly related.

H_1 → (t,c) → " " " " " are significantly related.

→ t : occurrence of term ; c → occurrence of class . Rank terms with :

$$X^2(D,t,c) = \sum_{e_t \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{(N_{e_t e_c} - E_{e_t e_c})^2}{E_{e_t e_c}}$$

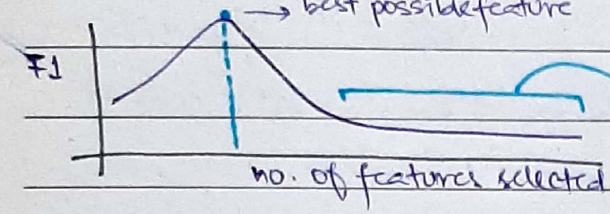
→ x^2 is a measure of how much expected counts E & observed count N deviate from each other. A high val. of x^2 indicates that the hypothesis of independence, which implies E & N are similar, is incorrect.

→ Example 13.4 for numerical.

Dazzle

Day / Date:

Feature Selection Graph



FS increases & at one point it reaches max. Then it decreases

redundant — less info features & irrelevant to our prob

effect of feature set size on accuracy

Feature Generation

The idea is to generate high level features (more abstract) from low level features.

Example: sentence dependency graph from sentence.

↓ i.e. If height & mass given so we can add them for calculating BMI.

Dazzle

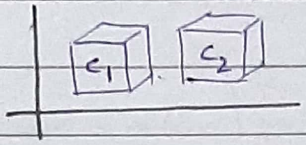
INFORMATION RETRIEVAL (pt. 2)

Day / Date: _____

VECTOR SPACE MODEL

- A doc is represented as a feature vector in feature-dimensional space.
- Distance is generally the notions of similarity for this model.
- General assumptions for classification:

Contiguity Hypothesis – Docs in same class form a contiguous region & regions of different classes don't overlap.



CLASSIFICATION – VSM

$D = \{D_1, D_2, \ldots d_n\}$

$C = \{c_1, c_2\}$

- Corpus – Classification Dataset
  - → Preprocessing
- Feature Selection & Weighing
- Similarity Function
- classification Algo
- Performance Evaluation

All docs of c_1 = □□□

f_1c_1 f_2c_1 → features of these docs

c_2 = □□

f_1c_2 f_2c_2

↓

create a vector space model of combined features

f_1	f_2	...

Two algos discussed:

① **ROCCHIO'S ALGORITHM** → can be used for text classification as well.

→ Divides the vector space into regions centered on centroids or prototypes, one for each class, computed as the centre of mass of all docs in the class.

→ Simple & an efficient classification, but inaccurate if classes are not approximately spheres with similar radii.

How algo works

→ Have class be doc to separate terms & merge to make one doc.

→ Make a high dimension space for each doc & calculate unit/centroid vector of all docs.

ie. $\vec{u_j} = \frac{\vec{d_1} + \vec{d_2} + \vec{d_3}}{3}$ → for each class calculate mean vector.

→ training func returns centroid of each class.

Dazzle

Day / Date: _____

In testing → calculate distance of the test doc with each mean (centroid) of training doc & jis se bhi least (minimum) distance hoga, that would be returned & that class would be assigned to the doc.

→ centroid vector ($\vec{\mu}_j$) is also called prototype.

② K-NEAREST NEIGHBOUR LEARNING

→ K-NN learning is an example based learning / instance based learning. Memory based method in which learning is just storing the representations of training eg. in D. (example to use neighbouring example se compare krenge in the class to see how similar they are. Jisse zyada similar wohi assign krdein gc).

→ Testing instance x: Compute similarity b/w x & all examples in D. Assign x the category of the most similar example in D.

→ It doesn't explicitly calculate a class/category prototype description.

* Algo for train & test in slides.

* example of numerical from book/slide. * k should be odd (preferable)

→ Advantages

- Simple, intuitive, easy to implement
- Only one hyperparameter ie. k

→ Disadvantages

- No training time, but large computation time for large datasets (more examples & more dimensions)
- lazy learner → no precompute model
- Sensitive to value of k, distance func & noisy data (example is placed in wrong class/ labelled wrongly or value of feature of example was changed) ← attribute noise, distance noise

Dazzle

Day / Date: _____

(Week 12) # TEXT CLUSTERING (Chp 16+17)

CLUSTERING — Unsupervised machine learning technique. It seperate out similar obj in a heterogeneous collection automatically.

→ The features of the obj implicitly identify for clustering.

→ An objective func is used to optimize similarity b/w obj in a cluster & reduce the similarity b/w different clusters.

*aik group ke beech mein (apas mein) similarity aur dou group ke beech major differences ho

↓ challenging prob → dual optimization → high Intra-cluster similar & low Inter-clust. similarity.

→ Doc clustering is a special prob → obj to cluster are docs.

- Natural lang Text
- Knowing exact no. of distinct groups (clusters)
- Producing an optimal clustering arrangement
- Label the groups
- Evaluation of clustering results.

CLASSIFICATION Vs. CLUSTERING

Classification → supervised learning → classes are human-defined & part of input to the leaning algorithm.

Clustering → Unsupervised → clusters are inferred from the data without human input

CLUSTER HYPOTHESIS

→ Docs in same cluster behave similarly w.r.t relevance to info need.

→ All applications of clustering in IR are based (directly/indirectly) on cluster hypothesis.

→ "closely ~~be~~ associated docs tend to be relevant to same requests".

Dazzle

Day / Date: _____

CLUSTERING APPLICATIONS (wide application in IR)

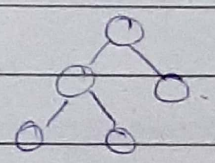
① Search result clustering ② Scatter-Gather ③ Collection clustering
④ Cluster based retrieval.

Scatter-Gather. → used for news refinement.
→ first scatter cluster (gather), the disperse (scatter) → into topics related to it,
The cluster & scatter again for more refined subtopics of user's interest.

FLAT Vs. HIERARCHICAL CLUSTERING

→ Flat Algorithms
→ clustering space is partitioned, partitions data (no relation) & treat all doc at same level.
↓ usually start with a random (partial) partitioning of docs into groups.
↓ refine iteratively ↓ into distinct/non-common groups (mutually exclusive) groups will be
Main algo → K-means

→ Hierarchical Algorithms



→ Creates a hierarchy of all possible cluster level.
→ Can be bottom up → agglomerative ; → can be top down → divisive
Main algo : HAC.

HARD Vs. SOFT CLUSTERING

Hard → Each doc belongs to exactly one cluster.
Soft → A doc can belong to more than one cluster.

K-MEAN CLUSTERING

→ Best algo. Simple, works well in many cases.
→ Used as a default / baseline for clustering docs.
→ Performance is determined by initialisation & appropriate distance measure.

Day / Date: _____

3 main steps in doc clustering:

① → Representation of docs ② → Similarity measure ③ → Clustering approach

Similarity func or measure

→ $\cos(d_i, d_j) = \frac{\vec{d_i} \cdot \vec{d_j}}{|d_i| \cdot |d_j|}$

measure → any calculation that is transformed in number. not necessary to have a unit.

metric → should have 3 properties
① Equal self similarity ② Minimality
③ Symmetry ④ Triangle inequality

Equal Self Similarity → $d(A,A) = d(B,B) \rightarrow \therefore s(A,A) = s(B,B)$

Minimality → $d(A,B) > d(A,A) \rightarrow \therefore s(A,B) < s(A,A)$

Symmetry → $d(A,B) = d(B,A) \rightarrow \therefore s(A,B) = s(B,A)$

Triangle Inequality → $d(A,B) + d(B,C) \geq d(A,C)$

where A & B & C are any doc assumed as obj.

K-mean: (Algo in book)

① Selects K random seeds (s_k) ← ① selects K docs

② Calculate centroid $\vec{\mu_k}$ of each cluster

③ And until cluster doesn't become same or distance is not minimum, we will compare every doc with cluster's mean ($\vec{\mu_k}$) & align min. with it (reassignment of vector) & recalculate mean ($\vec{\mu_k}$) → recomputation of centroids.

→ 3 criterion to stop clustering in k-mean:

① → arrangement is consistent. ② Give specific no. of iteration eg. i=5. (fix the loop)

③ → can stop it on the some specific value of centroid.

→ **Variants of K-means**

- **K-medoids**: resistance to noise/outliers b/c works using median.
- **K-modes**: extension to categorical data clustering analysis
- **CLARA**: extension to deal with large datasets

Dazzle

Day / Date: _____

CONVERGENCE of K-MEAN

→ first there are almost K^N ways to partition N data points into k

For each iteration of algo, we produce a new clustering based only on old clustering.

→ If old clustering is same as new, then next clustering will again be same.

→ If new clustering is different from old, then newer one has a lower cost.

→ Since algo iterates a function whose domain is a finite set → ∴ the iteration must eventually enter a cycle. The cycle must have length exactly 1. Hence k-means converges in a finite no. of iterations.

* N is generally a very greater no. as compared to K. ($N >>> K$)

There are K^N clusters. & running time of K-mean → $\underbrace{K * N}_{\text{time is bounded on no. of docs}} \propto |N|$

(partitions)

DRAWBACKS OF K-MEAN

① Sensitive to initial seeds (agr seeds dur hui ya outlier hui)

② Difficult to compare results

③ Fixed no. of clusters, difficult to predict the actual no. from datasets.

EVALUATING CLUSTERING TASK

→ The task is very challenging ∴ evaluation is also challenging.

→ Evaluation of 2-types:

Internal — Only data is used to evaluate the clustering result

(When data is not labelled, and we don't have any way to evaluate clustering done, so we calculate distance of obj among a cluster & distance b/w clusters. If obj among cluster are tightly packed (less distance) & distance b/w cluster is greater. It means 'good clustering'

Day / Date: _____

External – Evaluation with gold standard (ground truth). (labelled data given) ~~but~~

→ we hide labels and cluster the docs. Then check if they are clustered acc. to their correct label. If 100% correct, then best clustering.

WHAT IS A GOOD CLUSTERING?

→ Internal criterion → • Intra class ie. Intra-cluster similarity is high.
• Inter-class similarity is low.

→ The measured quality of a clustering depends on both the doc representation & similarity measure used.

→ External criterion → clustering should reproduce the classes in gold standard

• First measure for how well we were able to reproduce the classes — Purity.

• Purity → tells how homogenous clusters are.

$$purity(\Omega, C) = \frac{1}{N} \sum_{k} \max_{j} |w_k \cap c_j|$$; should be in range [0,1]
0 → not homogenous
1 → max. u

$\Omega = \{w_1, \ldots w_K\}$ → set of clusters, $C = \{c_1, \ldots c_n\}$ → set of classes.

for each cluster w_K → find class c_j with most member n_{kj} in w_K.

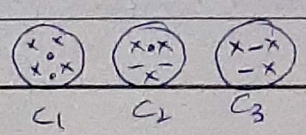
N = total no. of points.

• Random Index (How many attempts are made to do 100% correct clustering → 2 if obj are picked randomly each time.

Assign 2 docs to same cluster if & only if they are similar.

→ true +ve (TP) decision assigns 2 similar doc to same cluster. True -ve (TN) " " "
dissimilar docs to different clusters.

→ 2 types of error we can commit. (FP) → assign 2 dissimilar doc to same cluster
(FN) → assigns 2 similar docs to different cluster.



→ 17 obj → so we have → ${}^{17}C_2 = 136$ ways to ~~make clusters~~ 2 obj at a time

→ Numerical example in book.

Dazzle

Day / Date: _____

- **Normalized Mutual Information (NMI)**

→ How much info does the clustering contain about the classification.

→ Singleton clusters (no. of cluster = no. of docs) have max MI → means totally homogeneous

→ one to one correspondence check of how much similarity with ground truth.

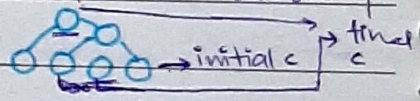
- relation classes with cluster.

- **F Measure**

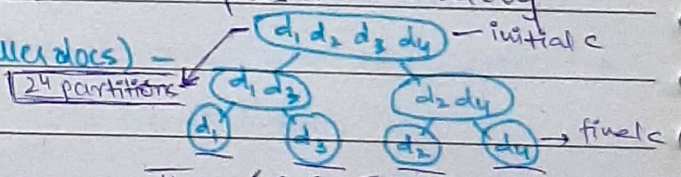
Precision & recall are weighted.

AGLOMMERATIVE VS. DIVISIVE CLUSTERING

→ Aglommerative (bottom up) method starts ~~with~~ each example in its own cluster & iteratively combine them to form larger & larger clusters.



→ Divisive (partitional, top-down) separate all examples immediately into cluster (cluster ko break kreinge in smaller docs)



- Simple HAC algo from book.

① Take out similarity ~~from~~ of each doc with one another (similarity of 2 instances)
→ starts with instances in a separate cluster & repeatedly

② Then merge krta hai maximum similar docs ko. To shrink no. of clusters until there is only one cluster.
→ The history of merging forms a binary tree or hierarchy.

~~Variations of HAC~~

Dendrogram

→ A binary tree that shows how clusters are merged/split hierarchically.

→ Each node on tree is a cluster; each leaf node is a singleton cluster.

→ computational complexity

- Running time of HAC if perfectly executed → $O(n^2)$. If not perfectly, can be: → $O(n^3)$ → $O(n^2 \log n)$

∴ data structure used matters a lot & affect efficiency of algo.

→ Different variations of HAC depending on the max function we are using.

→ In order to maintain an allover $O(n^2)$ performance, computing similarity to each cluster must be done in constant time.

Dazzle

Day / Date: _____

→ Variations of HAC

① Single Link Clustering (aka connectedness, the minimum one nearest neighbor method)

→ considers distance b/w 2 clusters to be equal to shortest distance from any member of one cluster to member of other cluster.

→ If data consist of similarities, the similarity b/w a pair of cluster is considered to be equal to the greatest similarity from any member of one cluster to member of other clust.

⇒ The distance b/w two clusters is represented by the dist. of 'closest pair of data obj' belonging to different clusters ie. $d_{min}(C_i, C_j) = \min_{p \in C_i, q \in C_j} d(p, q)$.

② Complete Link Clustering (aka diameter, the maximum method or farthest neighbour method)

→ considers method the distance b/w two clusters to be equal to longest distance from any member of one cluster to member of other cluster.

⇒ farthest pair of data obj ⇝ $d_{min}(C_i, C_j) = \max_{p \in C_i, q \in C_j} d(p, q)$ → max distance & min. similarity b/w those points

③ Average Link Clustering (aka minimum variance method)

→ considers distance b/w two clusters to be equal to avg distance from any member of one cluster to any member of other cluster.

⇒ avg dist. of all pairs of data obj ⇝ $d_{min}(C_i, C_j) = \text{avg}_{p \in C_i, q \in C_j} d(p, q)$

④ Centroid Distance

The distance b/w two clusters is represented by distance b/w the means of clusters.

$d_{mean}(C_i, C_j) = d(m_i, m_j)$ where (m_i & m_j are means of c_i & c_j)

→ There are many (20-25) variations. All variations will give different clusters.

Advantage of HAC → we don't have to predefine K → how many clusters we want.

• once clustering is done, we can then extract K no. of clusters from it.

Dazzle

Day / Date: _____

WEB SEARCH BASIC (Chp 19)

Web Search - Client Server

Client — generally a browser — an app with GUI (environment)

Server — the server communication with a client via a protocol HTTP.

→ lightweight, simple & asynchronously carrying a variety of payloads (text, img) encoded in a simple markup lang. called HTML.

→ resources & webpgs are available & we browse it by inputting URL.

- HTTP be like as a responder → server ; - HTTP be like as a requester → client.

HTTP

- An app protocol for distributed, collaborative & hypermedia info.
- HTTP header contains a lot of fields for effective transfer of info.
- Persistance connection — doesn't let connection die. It was client/server connection & things are transferred b/w it.

→ **Major Advancement**

- Session state can be maintained — session info (client-server) can be served through cookies.
- Support authentication ~~method~~ mechanism.

→ **HTTP Status Code**

Client sends a GET url command then <* blank lines *> & then Carriage Return (CR), line Field (LF) (CRLF)

↳ (protocol command completed)

In return, server retrieves a pg & forwards index.HTML pg to client.

- Informational 1XX
- Redirection 3XX
- Server Error 5XX
- Successful 2XX
- Client error 4XX

Dazzle

Day / Date: _____

Server Side Scripting

Client pe time continuously server se fetch hota ke change hota raha hotta hai (A no. of server side scripting available).

Client side Scripting

Generally UI & interaction with local machine, mostly Java Script.

Cascading Style Sheet (CSS)

A lang. that describes the style of an HTML doc. ; → depending on browser displays output differently on a single webpg.

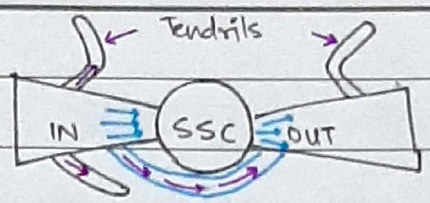
* Server side pe data fixed nai. can make new pgs from each clients' request & render it acc. to user.

WEB AS A GRAPH

- A web can be viewed as a graph of connected web pgs.
- Anchor text & link to a web resource is used to link these pgs.
  - ↓ jo link ke upar text hotta hai — resource info.
- Outlink → hyperlink out of a pg.
- Inlink → hyperlink into a pg.
- BOW TIE MODEL
  - → First pg (home pg. is known usually)
  - → Homepg → Inpage → (goes Through scg of pg SSC (round) & then you reach outpg.
    - * we can go back from out To' SSC.
    - ↓ (Surfing Through different web pg)
  - → There's a tube of → classified channel (dark/classified ~~dark~~ web pgs) → not all users can access (pornography, data selling)
  - → Tendrils → incompatible / not known
  - → Har pg. mein i links hottay hain & they grow exponentially on avg.
    - ↓ Power law states that these inlinks & outlinks are govern with a power functions. In link proportional to $1/a^i$.

Dazzle

Day / Date: _____



SSC → highly connected pgs. that surf you among pgs.

Tendrils → dead end web pgs → jahan se return nai kr sakte ; can only be accessed by only few people — not everyone allowed.

Web as a Graph

- Spam (in context of web search) is the manipulation of web pg content for the purpose of appearing up high in search result for selected keywords.
- Web pgs rank highly on selected keywords.
- Adversial Info Retrieval is the area of study making a ~~bb~~ balance b/w SEO & SE
  → computator ki info build using search engineering.

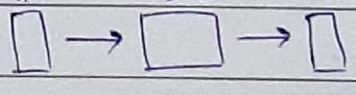
SEO → to provide consultancy services for clients who seek to have their web pgs ranked on selected keywords.

IN → home pg → initial

SCC → tightly coupled pgs → they move us to OUT

→ Stochastic Model (a parallel model)

Prob of opening a link



→ Adversarial IR : Area of study making a balance b/w SE & SEO

Organic → Free web/content only Paid → Adds advertisement too

Advertising as an Economic Model

→ Ad words are located in query & the ad are pushed to make some business value.

→ Typically, these ads are priced on cost per mile (CPM).

→ cost per click (CPC) → cost per user action (CPA)

→ Understanding how search engines do the ranking & how to allocate marketing campaign budgets to different keywords & to different sponsored search engines for maximum (SEM)

→ click Spam — a phenomenon for advertising competition (SEM)

Dazzle

Day / Date: _____

- Pure vs. Sponsored : Pure — only displaying content of web, not paid extra content (ads)

  Mix — Pure + Sponsored — ads + content

- Combining Pure vs. Sponsored
- Blog vs. Product reviews

USER NEEDS

- Informational — want to learn about something (40–65%) eg. low bp
- Navigational — want to go to some page (25–15%) eg. PIA
- Transactional — want to do something (35–20%) • Access service, download, shop
- Gray areas — find a good hub : exploratory search 'see what's there'. • car rental brazil

WEB INFORMATION DISCOVERY

→ Directories

- Taxonomies populated with web pgs in categories eg. Yahoo!
- The user to browse through a hierarchical tree of category label.

→ Search Engine

- Full Index search such as Infoseek.
- The user with a keyword search interface supported by inverted indexes & ranking mechanism.

Web Characteristics

① Web User Interaction ② Web as a Graph ③ Web Spam

↓

Anchor text (along with link)

Top 10 Search Engines

① Google ② Mahalo ③ Yahoo ④ Bing

Index Size & Estimate

→ Capture/Recapture Method - Pick a random page for index of E1 & check/test whether it

Day / Date: _____

is in E2's index & vice versa. These experiments give us fraction x & y such that x frac. of pgs in E1 are in E2 while y frac. of pgs in E2 are in E1.

→ The |Ei| denotes the size of index of search engine Ei → ∴ x|E1| ≅ y|E2|.

(Ak search engine ke muqabley mein kis 2 engines mein se konsa engine pe zyada data index hua raha)

$\frac{|E1|}{|E2|} \approx y/x$ → konsa engine zyada pgs index kr raha.

Sampling Methods (How to choose pgs for testing)

① Random searches (randomly selected ; have biasness)

② Random IP addresses

③ Random walks

④ Random queries

Duplicate / Near Duplicate Detection

→ While indexing, we may come up for duplicate. Checksum is common method to detect a duplicate & (Full duplicate have 90–100% similar data)

Near duplicate — not identical, but a portion is common, based on preset threshold we can filter out the near duplicates (50% + content same)

Shingling — Given a +ve integer k & a seq. of terms in doc d, k-shingling is a set of all consecutive seq. of k terms in d.

used to find near duplicate.

eg. A = a rose is a rose is a rose

shingling set 2 = { a rose is a ; rose is a rose ; is a rose is .. } ; |2| = 5.

Jaccard = 2/5 b/w |A| = 4 (took 4 terms at a time, then next 4).

→ Duplicate can compairtively be easily found by checksum, whereas it is challenging to find near duplicate.

Dazzle

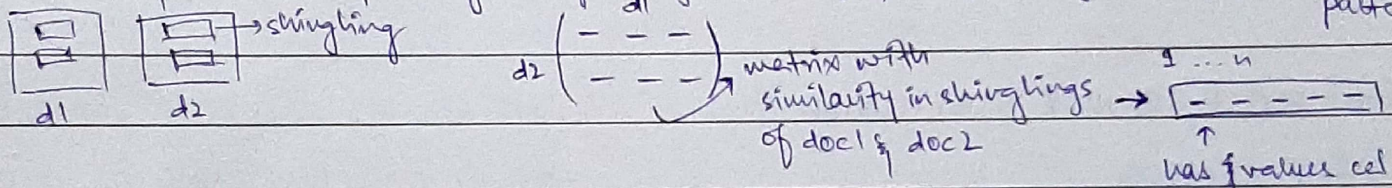
Day / Date: _____

Method → calculate shinglings of each pg/doc. The put them in a hash-table & calculate similarity b/w them to see if the pgs/doc are near duplicates or not.

Doorway Pages → these pgs are used for Search engine optimization.

→ Near Duplicate Scaled Approach

- A pair wise approach — using shingling overlap to detect near duplicates.
- We can perform better by using a large integer hashing func & doing hashing for shingling pattern.



→ next time jab bhi koi similar pair aye ga, we can easily search in hash-table.

shingling

→ we need hashing of minimum 2^{64} (High computational cost)

Important Topic

→ User Needs (Query types)

→ Index Size & estimate

→ Duplicate & near duplicate

Dazzle

Day / Date: _____

WEB CRAWLER (Chp 20)

→ Web crawling is the process by which we gather pgs from the web to index them & support a search engine.

→ Objective of crawling is to quickly & efficiently gather as many useful web pgs as possible, together with the link structure that interconnects them.

→ web crawler is sometimes referred to as Spider

Steps * process starts by ① URL queue. then url generates ② a DNS & send to exact ③ IP addresses. this is a two way communication.

FEATURES A CRAWLER MUST PROVIDE

① ROBUSTNESS

The crawler must be robust to deal with a large no. of linked pgs from a website. Sometimes a server traps a crawler, and the crawler is stuck b/w giving two pgs. ∴ The crawler must identify these traps.

② POLITENESS

Web servers have both implicit & explicit policies regulating the rate at which a crawler can visit them. These politeness policies must be respected.

③ DISTRIBUTED

The crawler should have the ability to execute in a distributed fashion across multiple machines.

④ SCALABLE

The crawler architecture should permit scaling up crawl rate by adding extra machines & bandwidth.

Dazzle

Day / Date: _____

(5) PERFORMANCE & EFFICIENCY

The crawl sys should make efficient use of various sys resources including processor, storage & network bandwidth.

(6) QUALITY

Given that a significant fraction of all web pgs are of poor ~~quality~~ utility for serving user query needs, the crawler should be biased toward fetching 'useful' pgs first.

(7) FRESHNESS

In many apps, crawler should operate in continuous mode. It should obtain fresh copies of previously fetched pgs.

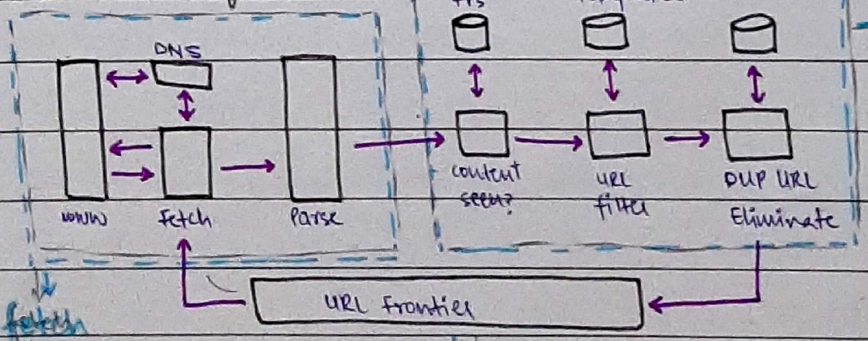
(8) EXTENSIBLE

Crawler should be designed to be extensible in many ways — to cope with new data formats, new fetch protocols & so on. Crawler archit. should be modular.

TYPES OF CRAWLER

- Path ascending crawler
- Topic focused crawler (only focused on certain content)
- Academic focused crawler (eg. google scholar)
- Semantic focused crawler

Architecture of Crawler



content decide

URL frontier → complex structure of queues

appending new URL

Dazzle

Day / Date: _____

URL frontier → containing URLs yet to be fetched in the current crawl.

At first, a seed set is stored in URL frontier, & a crawler begins by taking a URL from the seed set.

DNS → domain name service resolution. Lookup IP address for domain names.

Fetch → generally use the http protocol to fetch the URL.

Parse → the pg is parsed. Texts (imgs, videos) and links are extracted.

Note: Read the chp from book for more details.

Day / Date: _____

LINK ANALYSIS (Chp 21)

WEB AS A GRAPH

Link analysis of web (as a graph) is based on following 2 assumptions:

① Anchor text pointing to pg B is a good description of pg B

② Hyperlink from A to B represent an endorsement of pg B by creator of pg A.

(⇒ This is not always the case) ie. many ~~hits~~ & links amongst pgs within a single website stem from the use of a common template.

↓ endoisement can be for a single domain.

CHALLENGES

- Pg might not provide an accurate description of itself.
- There is a gap b/w terms in a web pg & how a web users would describe web pgs. (terms don't define content of web pg)
- Web pgs is a composition of text, graphics & img. Standard IR approach doesn't support searching with these rich contents.
- Anchor text along with extended info from anchor extended gives a good idea of web pgs.

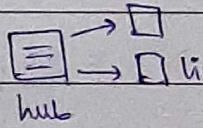
* Anchor text means the text along with a link (description of a link of pg) & extended anchor text means a little content from above & below anchor text.

HITS (Hyperlink-Induced Topic Search)

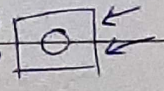
→ Also known as hubs & authorities is a link analysis algo that rates ~~pg~~ web pages.

• To calculate/decide the score of each pg; hub score & authority score.

hub → pg that mainly contains outlinks (pgs it is referring)



Authority → the web pg is the best link of a domain



most of referred link. other pgs have used it as an inlink. other pgs (hub) are referring to this pg.

Dazzle

Day / Date: _____

→ A good hub represents a pg that pointed to many other pgs

→ A good authority represents a pg that was linked by many different hubs.

* The algo assigns two scores for each pg; authority & hub value.

Authority → estimates the value of the content of the pg.

Hub → estimates the value of its links to other pgs

HITS Algorithm

① Given a search query Q, collect a subset of 200 top webpgs as input (seed) that contain highest frequency of query Q.

② Add the collection of web pgs that points or are pointed by these top 200 webpgs.

Create adjacency matrix A among these web pgs.

③ Initialize the hub & authority column vectors U & V with values 1.

④ For a set k no. of iterations, do the following:

(i) Update the authority scores through authority metrix V.

(ii) Update the hub scores through hub metrix U.

(iii) Normalize the hub metrix and authority metrix U & V.

⑤ Rank the webpgs acc. to the authority score as reflected through authority metrix V.

example (in video)

a → authority score , h → hub score

$a^{(1)} = A^T \cdot h^{(0)}$

$h^{(1)} = A \cdot (a)^0$

$h^{(0)} = a^{(0)} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$ or $\begin{bmatrix} 1/n \\ 1/n \\ 1/n \\ 1/n \end{bmatrix}$

$h^{(0)}$ → hub score at iteration 0.

for eg. $a^2 = A^T \cdot h^{(1)}$ → one previous state

Day / Date: _____

$$\begin{bmatrix}1\\3\\5\\1\\2\\1\end{bmatrix} = \begin{bmatrix}1/(1^2+3^2+5^2+1^2+2^2+1^2)\\3/(1^2+3^2+\dots)\\\vdots\end{bmatrix}$$

normalized

→ Authority score zyada tou most authority pgs.

→ ∀p we update auth(p) to be summation :

$$auth(p) = \sum_{i=1}^{n} hub(i)$$

; n → total no. of pgs connected to p
i → a pg connected to p

$$hub(p) = \sum_{i=1}^{n} auth(i)$$

matrix multiplication

ISSUES of HITS

① → expensive in sense of computing time.
② → Query dependant
③ → Executed at query time, not at indexing time.
④ → Not commonly used by search engines
⑤ → It computes 2 scores, not 1.
⑥ → It is processed on a small subset of relevant documents (a focused subgraph / baseset), not all docs as was the case in pg rank algo.

* With the help of query expansion, we can find other relevant & related docs to the query term.

Note: See numerical at end of slides.

Dazzle

h^0 or a^0 $\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ $[1\ 1\ 1]^0$ • $a' = A^T \cdot h^0$ → first iteration of authority score
all 1's • $h' = A \cdot a^0$ → " " " hub score.

Day / Date: _____ eg. $A^2 = A^T \cdot h'$

Query extension is done so that
pt includes the pgs that are pointing towards page that are ranked by query.
In a lot of cases anchor text is not very clear, so if query term is not present, we want to include those pages too; those pgs which are tightly knitted to the pgs retrieved (either in a authority way or a hub way)

pehlay query pe filter kiya that those pgs that ~~are~~ have high freq. of query term.

PAGE RANK

→ no. of web pgs → N
→ Anchor text → describes link & have hyperlink URL
- forward link p1 $\xrightarrow{to}$ p2 → Outlink
- backward link p1 $\xleftarrow{by}$ p2 → Inlink

Page Rank is also a link analysis algo
assigns a numerical weighting to each element of a hyperlinked set of doc → Har pg. ki importance btata hai pg. rank.

Damping fac → ait pg. pe pohanch jaye in without any link. Prob. that a user opens a new web pg to begin random walk.

Day / Date: _____ Summers.

PR(P) → pg rank of pg. P

$deg(P)^-$ → ☐←

$deg(P)^+$ → ↘☐↙

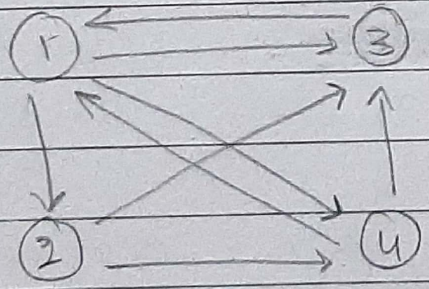
$N(P)^-$ → 1

$N(P)^+$ → 2

W → hyperlink matrix representing network whose entries constitute the fractional PageRank contributions.

w → eigenvector containing ranks for each vertex in the network.

d is kept from 0.1 – 0.2. And usually d = 0.15



$PR(1) = \frac{2}{3}$

P

$\gamma_a = \frac{\gamma_4}{2}$ $\gamma_y = \frac{\gamma_4}{2} + \frac{\gamma_a}{2}$

$= \frac{\gamma_4}{2} + \frac{\gamma_4}{2} \times \frac{1}{2}$

$\gamma_4 = \frac{2}{2}\gamma_4 \times \frac{1}{2}$

$\gamma_y = \frac{\gamma_y}{2} + \frac{\gamma_a}{2}$

$\gamma_y = \frac{\gamma_y}{2} + \frac{\frac{\gamma_y}{2}}{2}$

$\frac{\gamma_4}{2} = \frac{\gamma_4}{2} \times \frac{1}{2}$

$\frac{\gamma_4}{2} = \frac{\gamma_4}{4}$

someday.

similar.

mornings.