Date_____

(9) and (10) Answer pg # 08 (disadvantages & challenges)

Chap #02 The term vocabulary and posting lists

* Tokenization → Process of chopping character streams into tokens. Linguistic preprocessing that deals with building equivalent class of tokens.

* Document processing pipeline
  ① Collect the documents to be indexed
  ② Tokenize the text
  ③ Do linguistic preprocessing of tokens
  ④ Index the documents that each term occurs in

* Challenges in Document processing:
  ① What character set is required for encoding
     eg: ASCII encoding for plain english text
     Multi-byte or single byte encoding scheme like UTF-8.
  ② What language the document is written in?
  ③ What format is it in?
     eg: pdf / html / word

nice

Date

entries to consider for a certain query term.
eg car search krenge to automobile ki posting list ka intersection bhi lega to cater synonyms.

② Perform the expansion during index construction.
eg: When a document contains 'automobile', we index it under 'car' as well.

Maintaining relations of unnormalized terms is less efficient because there are more postings to store & merge. These expansions are asymmetric.

- Normalizing Tokens to remove diacritics.

- Case folding : Reduce all letters to lower case.

* Morphological Analysis: (Forms of things)

① Inflections: Adding suffix to a word, that doesn't change its grammatical category
eg: Tenses in verbs
Plural in nouns.

② Derivations: Adding suffix to a word that changes its grammatical category.
eg: Noun → adjective → verb.

Date

Dictionary	Thesaurus
① Contains an alphabetical list of words including meanings, etymology and pronunciation	① List of words with synonyms and antonyms, arranged either alphabetically or thematically
② It defines a word.	② It gives a list of words with similar meanings.
③ Some dictionaries use the International Phonetic Alphabet	③ There are various thesauri, some of which may give additional info like phrases, sentences and clauses.

Stemming	Lemmatization.
① Refers to crude heuristic process that chops off the ends of words.	① Refers to chopping properly with the use of vocabulary & morphological analysis of words.
② Operates on a word without knowledge of the context.	② Removes inflection endings & returns the base word.

Date

* Normalization:

- We need to normalize words in documents as well as in queries, so that the tokens in the query exactly match the tokens in the token list of the document.

- Token normalization is a process of canonicalizing tokens so that matches occur despite superficial differences in the character sequences of the tokens.

- The most standard way is to implicitly create equivalence classes of terms

Method 1

eg: deleting periods to form a term

U.S.A — USA

eg: deleting hyphens

anti-discriminatory, antidiscriminatory.

~~- Normalizing tokens to remove diacritics~~

~~- Case folding: Reduce all letters to lower case~~

→ Synonyms to handle Rine K Nye

Method 2

- Another way of normalization is to maintain relations between unnormalized tokens.

This can be done in two ways;

① Index unnormalized tokens and maintain a query expansion list of multiple vocabulary

Date

④ Choosing a document unit. Selecting too large or too small document affect precision and recall.

⑤ Tokenization is challenging.
- Difficult to cater hypheration, apostrophe
- Compound words
- Abbreviation, numbers, phone-number etc.
- language identification.
eg: splitting on basis of white spaces is easier for english language only. Challenges arise while dealing with other languages.
- Splitting on white space can also split what should be regarded as single token
eg: San Francisco.

* Stop words :-
- Extremely common words are called stop words.
- We usually drop stop words entirely from dictionary since they have little impor-tance and occur very frequently.
- But modern technique do not allow to throw any thing from documents.
- Stop words play important role in some special queries like phrase queries, relational queries, title of songs etc

Date_____

(3) Stemming is faster, easier to implement and reduces accuracy.	(3) Lemmatization is slower, but accurate.
(4) Example: Operational	(4) Example : Operational
Result : Oper, O, Op etc.	Result : Operate.

Conclusion: The goal of both stemming & lemmatization is to reduce inflection forms & some times derivationally related forms of word to a common base form.

→ Porter Stammer.

Consist of 5 phases of word reduction, applied sequentially.

Within each phase - there are various conventions to select rules.

See slides 15 - 18

Date

* Faster postings list intersection via skip pointers:

- If the postings list lengths are m and n, the intersection takes O(m+n) operations.
- Can we do intersection in sub linear time?
- We can, if index isn't changing too fast
  i.e using skip list.

Skip List:
- Augmenting postings list with skip pointers forms skip list.
- Skip pointers are shortcuts that allow us to avoid processing parts of the postings list that will not figure in the search result.
- Where to place skip pointers?
- More skips means;
  shorter skip span
  lots of comparisons to skip pointers.
  lots of space storing skip pointers.
- Fewer skips means
  long skip span
  Fewer comparisons
  Fewer opportunities to skip.
- Skip pointers are easy to implement, if indices are static.
- Only useful for 'AND conjunctive' queries.

nice

Date

Q Why skip pointers are not useful for disjunctive queries (x OR y) ?

Because in queries of form "x OR y", it is essential to visit every docID in the postings lists of either terms, thus no need for skip pointers

* Phrase Queries
  - We want to be able to answer queries in the form of ~~queries~~ phrases like Stanford University.
  - Simple postings lists require post processing to eliminate false positive results reducing efficiency.

* Solution 1 to Phrase Queries: Biword indexes
  - Index every consecutive pair of terms in the text as phrase.
  - Each of these biwords is now a dictionary term.
  - Two-word phrase query processing is now immediate
  - Longer phrases can be processed by breaking phrases into conjunctive biword boolean query.

eg: Stanford university palo alto
"Stanford university" AND "University Palo" and "Palo Alto".
Results in False positives.

nice

Date

Extended biwords:
- Tokenize the text & perform parts of speech tagging.
- Group terms into nouns (N) and function words (X)
- Any string of form NX*N called extended biword.

Issues:
- False positives
- Index blow up due to bigger dictionary.

* Solution 2 to Phrase queries: Positional indexes
- In the postings, store for each term the position(s) in which tokens of it appear:

| term | doc-frequency | → | D_1: Position 1, 2, 3 | → | D_2: Position 4,5 |

Term

eg < hello : 200; doc frequency
docid - 1: 9, 11, 14, ... ; positions
2: 3, 4 ...;
3: 6, 7, 8 ...;
⋮
n: . - - .; >

- Can handle proximity queries.

nice

Date

eg to be or not to be

- Extract inverted index entries for each distinct term. i e : to, be, or, not.
- Look for documents containing 'to' and 'be'
- Look position of 'to' in a document, then look if position of 'be' occur one higher than position of 'to'. This means both terms are adjacent
- Process all terms in a similar manner.

* Proximity Queries:
  - Have k words in between w_i and w_j
  - /k mean "within k words of".
  - Only positional indexes can be used for such queries

* Positional index size:
  - Positional index expands postings storage
  - Useful because phrase & proximity queries can be handled easily.
  - Need an entry for each occurence in a document.
  - Index size depends on avg document size.

nice

Date

* Combination Schemes:
  - It is a combination of biword indexes and positional indexes.
  - For queries like 'Michael Jackson' it is inefficient to keep on merging postings lists
  - This technique uses biword indexes for certain queries & positional indexes for other phrase queries based on different criteria.
  - Gives speed up

Next word index → Another scheme

< Food for thoughts > chap # 02

① Answer pg # 12
② Answer pg # 12 & 13
③ Answer pg # 16 & 17

④ (a) No. of comparisons = 11
Compares till we reach 47 in upper postings list.

(b) No of comparisons = 6.
list 1 → skip pointer of length = $\sqrt{16}$ = 4
list 2 → 4 4 4 4 = 1

⑤ Answer pg # 20

Date

© - Positional index extracts inverted index for each distinct w_i and w_j.
- look for document 'd' containing w_i at index m and w_j at index $m+k$, where k is equals to 1 for phrase query

eg Q_1 = hello world

<hello : 10 ;	<world : 5 ;
d_1 : 1, 2, 3 ;	d_1 : 4, 6, 10, 11 ;
d_2 : 4, 6, 8, 12 ;	d_2 : 5 ; >
d_3 : 7, 8, 9 ; >	

Output :
d_1 b/c hello at position 3 & world at position 4
d_2 b/c hello at position 4 & world at position 5.

① pg # 22

nice