Total No. of Questions – [04] Total No. of Printed Pages:04

Marking Scheme [P118-134 (T1)]

OCTOBER 2018 / IN - SEM (T1)

F. Y. M. TECH. (Computer Engineering) (SEMESTER - I)

COURSE NAME: Program Elective II

(Information Retrieval and Web Mining)

COURSE CODE: CSPA11184A

(PATTERN 2018)

Q.1	a.	**Lossy and lossless index compression techniques** 1. Lossless compression: All information is preserved. a. What we mostly do in IR. 2. Lossy compression: Discard some information 3. Several of the preprocessing steps can be viewed as lossy compression: case folding, stop words, stemming, number elimination. 4. Dictionary Compression • Dictionary-as-a-String • Blocking • Front coding 5. Posting List Compression • Variable length encoding • Variable Byte (VB) codes	[3+3]
	b.	**Signature files in IR** Characteristics : • Word-oriented index structures based on hashing • Low overhead (10%~20% over the text size) at the cost of forcing a sequential search over the index • Suitable for not very large texts • Inverted files outperform signature files for most applications Structure :	[2+2]

- Use *superimposed coding* to create signature.
- Each *text* is divided into logical blocks.
- A *block* contains *n* distinct non-common words.
- Each *word* yields "word signature".
- A *word signature* is a *B*-bit pattern, with *m* 1-bit.
    - Each word is divided into successive, overlapping triplets.
      e.g. *free* --> □*fr, fre, ree, ee* □
    - Each such triplet is hashed to a bit position.
- The word signatures are OR'ed to form *block signature*.
- Block signatures are concatenated to form the *document signature*.

Example

1 **Example (n=2, B=12, m=4)**

word			*signature*	
free	*001*	*000*	*110*	*010*
text	*000*	*010*	*101*	*001*
block signature	*001*	*010*	*111*	*011*

Search

- Use hash function to determine the m 1-bit positions.
- Examine each block signature for 1's bit positions that the signature of the search word has a 1

Q.2	a.	**Index construction** The inverted index of a document collection is basically a data structure that attaches each distinctive term with a list of all documents that contains the term. **Different techniques** 1. BSBI (Blocked Sort-Based Indexing) algorithm 2. SPIMI (Single-pass in-memory indexing) algorithm 3. Distributed indexing 4. Dynamic indexing	[2+4]
	b.	**Four challenges of unstructured data/text** • No stable document collection (spider, crawler) • Invalid document, duplication, etc.	[1+1+ 1+1]

		• Huge number of documents (partial collection) • Multimedia documents • Great variation of document quality • Multilingual problem • Vocabularies mismatching ▪ Synonymy: e.g. car v.s. automobile ▪ Polysemy: table • Queries are ambiguous, they are partial specification of user's need • Content representation may be inadequate and incomplete	
Q.3	a.	**Three models of Language Models for IR** A language modelis a probabilistic mechanism for generating text •Language models estimate the probability distribution of various natural language phenomena –sentences, utterances, queries Language Modeling Techniques • •N-grams • •Class-based N-grams • •Probabilistic CFGs • •Decision Tree	[2+2+2]
	b.	**Two types of Query expansion** 1. Global Analysis: (static; of all documents in collection) a. Controlled vocabulary i. Maintained by editors (e.g., medline) b. Manual thesaurus i. E.g. MedLine: physician, syn: doc, doctor, MD, medico c. Automatically derived thesaurus i. (co-occurrence statistics) d. Refinements based on query log mining i. Common on the web 2. Local Analysis: (dynamic) a. Analysis of documents in result set	[2+2]

Q.4	a.	**Latent Semantic Indexing** Suppose that we use the term frequency as term weights and query weights. The following document indexing rules are also used: • stop words were not ignored • text was tokenized and lowercased • no stemming was used • terms were sorted alphabetically. **Solved Example**	[2+4]
	b.	**Ad hoc Retrieval** • Text-based retrieval • Given a query and a corpus, find the relevant items • query: textual description of information need • corpus: a collection of textual documents • relevance: satisfaction of the user's information need • "Ad-hoc" because the number of possible queries is huge **Pseudo relevance feedback** • The user issues a (short, simple) query. • The search engine returns a set of documents. • User marks some docs as relevant, some as nonrelevant. • Search engine computes a new representation of the information need. Hope: better than the initial query. • Search engine runs new query and returns new results. • New results have (hopefully) better recall. • Provides a method for automatic local analysis • Pseudo-relevance feedback automates the "manual" part of true relevance feedback. • Pseudo-relevance algorithm: • Retrieve a ranked list of hits for the user's query • Assume that the top *k* documents are relevant. • Do relevance feedback (e.g., Rocchio) • Works very well on average • But can go horribly wrong for some queries. • Several iterations can cause *query drift*.	[2+2]