National University of Computer & Emerging Sciences

Midterm Examination – Fall 2017 (Sol)

Course: IR&TM (CS567) Time Allowed: 1 Hour
Date: September 18, 2017 Max. Marks: 40

Instructions: Attempt all question. Be to the point. Draw neat and clean diagram/code where necessary. Answer each question on the new page of the answer book, no marks for junk explanations. You must address all inquires in a question.

Question No. 1 **[Time: 25 Min] [Marks: 20]**

Answer the following questions briefly using 4-5 lines of answer book. Be precise, accurate and to the point, only answer genuine query in the question. Each question is of 2 marks.

a. Define what do we mean by Token in IR preprocessing of resources.

A token is an instance of a sequence of characters in some particular document that are grouped together as a useful semantic unit for processing.

b. What are some of the drawbacks of Vector-Space Model (VSM)?

- The original order of words from the document are lost in VSM representation.
- Theoretically assume terms are independent with each other.
- Weighting schemes are intuitive for not very formal

c. The term frequency* inverse document frequency (TF*IDF) weighting scheme is considered the best in IR. How weights are related to occurrence of term (statistics) in this scheme?

The tf-idf weighting scheme assigns term t a weight in document d that is
1. highest when t occurs many times within a small number of documents (thus lending high discriminating power to those documents);
2. lower when the term occurs fewer times in a document, or occurs in many documents (thus offering a less pronounced relevance signal);
3. lowest when the term occurs in virtually all documents.

d. What is the idf of a term that occurs in every document? Compare this with the use of stop word lists.

If the term occurs in every document than df=N hence (IDF = log D/df = log N/N = log 1 = 0)
whereas, for stop-word list, we simply do not consider these words for vector space model hence they do not have any contribution in tf*idf weighting.

e. Define what do we mean by leading wildcard query? Give an example. Suggest a data structures that best handle such queries.

When use is not sure about prefix-string of a single query string, it tries to formulate a leading wildcard query, by placing a * in place of prefix. For Example: a query such as *mon is a leading wildcard query. A reverse B-Tree (B-tree corresponds to a term in the dictionary written backwards) can handle such queries very well. A walk from the root corresponds to all terms given a prefix.

f. How permuterm index is used to process trailing wildcard queries? Explain from the start of identifying the possible terms, to access posting list and to finally get the result set.

The trailing wildcard query is of the form car*; In a permuterm index each dictionary entry is index by placing a $ character at the end, the string so formed is rotated single character at a time to produce all permuterm vocabulary. For example, card will be like card$, ard$c, rdca, dcar, $card and for a query like car* we will search for *$car and will get all term start with a prefix car. The posting list for all such term will be union to get the result.

g. Outline three differences between stemming vs lemmatization.

Stemming	**Lemmatization**
It is a heuristic- rule based approach, generally fast and use a single term. It generates unreadable tokens.	It is a rigor process that uses a dictionary and uses context to determine the lemma, considered a slow approach. It generates readable lexeme from the dictionary.

h. In Chinese, while using characters of mainland china, there is no whitespace between words, not even between sentences. what are the challenges for a tokenizer while processing text from such languages?

In these language a major problem with tokenizer is to identify word boundaries. This is called word segmentation. Methods of word segmentation vary from having a large vocabulary and taking the longest vocabulary match with some heuristics for unknown words to the use of machine learning sequence models, such as hidden Markov models or conditional random fields, trained over hand-segmented words.

i. What do we mean by context sensitive spelling corrections? Give an example of spelling error of this kind?

Context sensitive spelling correction try to correct a longer phrase of sentence by analyzing its surrounding context-words. A query of the form "flew form Paris" can easy be identified as contextual error as form is a mismatch with context to flew and Paris. Isolated word correction usually failed as all individual words are correct. Context sensitive spelling correction would suggest it as "flew from Paris".

j. Can the tf-idf weight of a term in a document exceed 1? Give an example.

Yes, it can be greater than 1. Consider a term appear 5 times, in document d1 for a collection containing only two documents d1 and d2. The tf*idf score of this term for doc1 will be 5* $\log_2(2/1) = 5$

Question No. 2 **[Time: 15 Min] [Marks: 10]**

Consider the following document collection consist of the four documents:

D={d1,d2,d3,d4}
d1= mary had little lamb
d2= little lamb mary had
d3= mary went with lamb
d4= lamb went with mary

1. Provide a term document matrix for the above collection.

	d1	d2	d3	d4
had	1	1	0	0
lamb	1	1	1	1
little	1	1	0	0
mary	1	1	1	1
went	0	0	1	1
with	0	0	1	1

2. Provide a term X term matrix for the above collection.

	had	lamb	little	mary	went	with
had	1	0	0	0	0	0
lamb	0	1	0	0	0	0
little	0	0	1	0	0	0
mary	0	0	0	1	0	0
went	0	0	0	0	1	0
with	0	0	0	0	0	1

3. On scale of (0-1) continuous real values, what is the similarity between pair (d1,d2) and (d3,d4) when you use Jaccard co-efficient of similarity.

Jaccard's Similarity (d1,d2) = 4/4=1.000
Jaccard's Similarity (d3,d4)= 4/4 =1.000

4. On scale of (0-1) continuous real values, what similarity score you give to pair (d1,d2) and (d3,d4) when you use semantic understanding of language.

Semantic Similarity (d1,d2) = 0.999
Semantic Similarity (d3,d4) = 0.999

Question No. 3 **[Time: 15 Min] [Marks: 10]**

Consider a corpus of three documents, which comprises of Vocabulary = {w_1 w_2 w_3 w_4 w_5 w_6 w_7}, assume that the subscript dictate order of dimension:

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}

Assume that we use TF= 1+ log (tf $_{t,d}$) for computing the term frequency of term t in document d. The Inverse document frequency (IDF) is define as log (N/df $_t$), where N=3 you can use TF*IDF to represent every term in VSM. Identify the ranking order of the documents with respect to query q.

	idf	tf-D1	tf-D2	tf-D3	tf-q	tf*idf D1	tf*idf D2	tf*idf D3	tf*idf q
w1	0.176	1	1	0	0	0.176	0.176	0	0
w2	0	1	1	0	0	0	0	0	0
w3	0.477	1.30103	0	1	0	0.620591	0	0.477	0
w4	0.176	1	1	0	1	0.176	0.176	0	0.176
w5	0.176	1	0	1	1	0.176	0	0.176	0.176
w6	0.176	0	1	1	0	0	0.176	0.176	0
w7	0.477	0	0	1	1	0	0	0.477	0.477

	Sim(D1,q)	Sim(D2,q)	Sim(D3,q)
w1	0	0	0
w2	0	0	0
w3	0	0	0
w4	0.030976	0.030976	0
w5	0.030976	0	0.030976
w6	0	0	0
w7	0	0	0.227529
	0.061952	**0.030976**	**0.258505**

Hence D3, D1 and D2 are the ranking for the given query.