Question Solutions for Week 5

Professor Yuefeng Li
School of Computer Science, Queensland University of Technology (QUT)

email y2.li@qut.edu.au

1. Abstract Model of Ranking

Text search or Information Retrieval (IR) is very different from traditional search tasks since it often uses an **inverted index** (a **special data structure**) that depends on a ranking function.

In the lecture notes, formally we define a ranking function R:

$$R(Q,D) = \sum_i (g_i(Q) * f_i(D)) \qquad (1)$$

for a given query Q and a document D.

Documents are written in natural human languages, which are difficult for computers to analyze directly. The popular method to represent a document is to select a set of document **features** (or **index terms**), where a document feature is some attribute of the document we can express numerically. For example, we can group document features into two categories: Topical features (e.g., important keywords) and Quality features (e.g., meta information, such as days since last update of the document).

In Eq (1), f_i is a kind of feature function that extracts a number from the document D (e.g., the keywork's frequency), and f_i is a similar feature function that extracts a value from the query Q. Each pair of functions g_i and f_i is multiplied together, and the results from all pairs are added to create a final document score R.

Please note in practice, the query features $g_i(Q)$ are mostly zero as Q is normally very short. This means that the sum for each document is only over the non-zero $g_i(Q)$ values.

2. Inverted Index

An inverted index is the computational equivalent of the index found in the back of a textbook. Normally, the book index is arranged in alphabetical order by index terms to help readers' search. Each index term is followed by a list of pages about that term or word.

Remember that our abstract model of ranking considers each document to be composed of features. With an inverted index, each word in the index corresponds to a document feature.

It is not hard to understand the data structures for designing inverted index in the lecture notes. The workshop for this part is to construct inverted index in python, and then perform the query process.

3. Query processing

Once an index is built, we need to process the data in it to produce query results. Even with simple algorithms, processing queries using an index is much faster than it is without one.

However, clever algorithms can boost query processing speed by ten to a hundred times over the simplest versions. There are two query processing techniques: document-at-a-time and term-at-a-time.

Question 1. Which of the following is wrong? and justify your answer.

(1) A document feature is some attribute of the document we can express numerically.
(2) A ranking function takes data from document features combined with the query and produces a score.
(3) Topical features are the only features we can find in documents.
(4) If a document gets a high score, this means that the system thinks that document is a good match for the query, whereas lower numbers mean that the system thinks the document is a poor match for the query.

Solution: (3)
A document also has Quality features (e.g., meta information, such as days since last update of the document).

Question 2. Which of the following is wrong? and justify your answer.

(1) The index is inverted because usually we think of words being a part of documents, but if we invert this idea, documents are associated with words.
(2) In an inverted index that contains only document information, i.e., the features are binary, meaning they are 1 if the document contains a term, 0 otherwise. This inverted index contains enough information to tell if the document contains the exact phrase “tropical fish”.

(3) Each index term has its own inverted list that holds the relevant data for that term. Each list entry is called a posting, and the part of the posting that refers to a specific document or location is often called a pointer.
(4) An extent is a contiguous region of a document. We can represent these extents using word positions.

Solution: (2)
Although a document that contains the words "tropical" and "fish" is likely to be relevant; however, we really want to know if the document contains the exact phrase "tropical fish". To do this, we need to add position information to the index.

Question 3. (Abstract Model of Ranking)

Assume the model of ranking contains a ranking function $R(Q, D)$, which compares each document with the query and computes a score. Those scores are then used to determine the final ranked list.

An alternate ranking model might contain a different kind of ranking function, $f(A, B, Q)$, where A and B are two different documents in the collection and Q is the query. When A should be ranked higher than B, $f(A, B, Q)$ evaluates to 1. When A should be ranked below B, $f(A, B, Q)$ evaluates to –1. If you have a ranking function $R(Q, D)$, show how you can use it in a system that requires one of the form $f(A, B, Q)$.

Solution:

For two documents A and B, we can define $f(A, B, Q) = 1$ if $R(Q, A) > R(Q, B)$; otherwise, if $R(Q, A) < R(Q, B)$, $f(A, B, Q) = -1$.

$$f(A,B,Q) = \begin{cases} 1, & if\ R(Q,A) > R(Q,B) \\ -1, & if\ R(Q,B) > R(Q,A) \end{cases}$$

For the special case of $R(Q, A) = R(Q, B)$, we may extend the definition to let $f(A, B, Q) = 0$, or assign 1 or -1 to $f(A, B, Q)$.

Question 4. (Query processing)
Inverted indexing is an efficient data structure to represent documents for information retrieval, where each index term is associated with an inverted list that contains a list of pairs of document number and count of the term occurrences in that document. The following table is an inverted index for 4 documents and their index terms.

and	1:1			
aquarium	3:1			
are	3:1	4:1		
around	1:1			
as	2:1			
both	1:1			
bright	3:1			
coloration	3:1	4:1		
derives	4:1			
due	3:1			
environments	1:1			
fish	1:2	2:3	3:2	4:2
fishkeepers	2:1			
found	1:1			
fresh	2:1			
freshwater	1:1	4:1		
from	4:1			
generally	4:1			
in	1:1	4:1		
include	1:1			
including	1:1			
iridescence	4:1			
marine	2:1			
often	2:1	3:1		
only	2:1			
pigmented	4:1			
popular	3:1			
refer	2:1			
referred	2:1			
requiring	2:1			
salt	1:1	4:1		
saltwater	2:1			
species	1:1			
term	2:1			
the	1:1	2:1		
their	3:1			
this	4:1			
those	2:1			
to	2:2	3:1		
tropical	1:2	2:2	3:1	
typically	4:1			
use	2:1			
water	1:1	2:1	4:1	
while	4:1			
with	2:1			
world	1:1			

Assume the abstract model of ranking is

$$R(Q, D) = \sum_i g_i(Q) f_i(D)$$

where f_i is the document topical feature function (the value of $f_i(D)$ is term t_i's counts in document D) and g_i is a query topical feature function ($g_i(Q) = 1$ if term t_i is in query Q; otherwise, $g_i(Q) = 0$). Let query Q = {**freshwater, fish, aquarium, tropical**}. Calculate each document's ranking score by using Term-at-a-time Algorithm. You are required to write the calculating process (or steps).

Solution:

```
L1 - freshwater      1:1 4:1
L2 - aquarium        3:1
L3 - fish            1:2 2:3 3:2 4:2
L4 - tropical        1:2 2:2 3:1

     L1 -            1:1            4:1
Partial scores       1:1            4:1
---------------------------------------------
Old Partial scores        1:1            4:1
     L2 -                      3:1
new Partial scores        1:1       3:1  4:1

---------------------------------------------
```

```
Old Partial scores       1:1       3:1  4:1
     L3 -                1:2  2:3  3:2  4:2
new Partial scores       1:3  2:3  3:3  4:3

--------------------------------------------
Old Partial scores       1:3  2:3  3:3  4:3
     L4 -                1:2  2:2  3:1
final scores             1:5  2:5  3:4  4:3
```