Tutorial for Virtual IR Lab

1. Overview

Virtual IR Lab is a web-based service for IR students to learn retrieval functions in a more interactive way. Specifically, it offers the following functionalities:

- ***Easy implementation of retrieval functions:*** Users are provided with a list of variables corresponding to the statistics retrieved from the index, and they only need to write a few lines of code to implement the retrieval function in a text box. The code will then be checked for any syntax errors.
- ***Flexible configuration of search engines:*** Users can configure/create a search engine by selecting a retrieval function and a test collection. Multiple search engines can be easily created and compared side by side.
- ***Evaluation:*** After creating a retrieval function, the users can evaluate its effectiveness over a few provided test collections by simply clicking a button. If a search engine is configured using an existing test collection with relevance judgments, the official queries and judgments will be displayed in the search engine interface so that users can analyze the search results to figure when their search engines fail or succeed.
- ***Leaderboard:*** The most effective retrieval functions for each collection will be displayed in the leaderboard. The users can conduct side-by-side comparison of their search engines with the best ones.

2. Quick Guide on Creating Retrieval Functions

The system computes the relevance score for each document at a time, and the relevance score of the document is stored in variable "score".

A simplest example of retrieval function can be shown as follows:

$$\mathrm{S}(\mathrm{d}, Q) = \sum_{w \in d \cap Q} c(w, d)$$

It says the relevance score of documents is computed as summing up the occurrence of all query terms in the document. We can implement this function as follows:

```
for(occur){
  score += tf[i];
}
```

Note that *for(occur)* means that we go through every query term that occurs in the current document, and *tf[i]* is the occurrences of term *i* in the current document.

Let us consider another retrieval function, Dirichlet Prior. Its function can be written as:

$$S(d,Q) = \sum_{w \in Q} \log \frac{c(w,d) + \mu * P(w \mid C)}{|d| + \mu}$$

We can implement this function in Virtual IR Lab as follows:

```
double dirMu=[1500 2000 2500];
for(all)
{
    score += log( (tf[i] + dirMu * termPro[i] ) / (docLength + dirMu) );
}
```

Note that *dirMu* is a variable that specifies the values for the parameter μ in the retrieval function, and we can specify multiple parameter values in the same time. *termPro[i]* is the query term probability in the collection, i.e., $P(w \mid C)$ and *docLength* is the document length, i.e., |d|.

Following is a summary of all the notations/variables you can use when implementing a retrieval function.

Variables	**Descriptions**
score	Relevance score of the current document, i.e,. S(d,Q)
for(occur)	Iterate through all query terms that occur in the document
for(all)	Iterate through all query terms
tf[i]	the occurrence of a query term i in the document, i.e., c(w,d)
DF[i]	the document frequency of term I, i.e., df(w)
qf[i]	the occurrence of query term i in the query, i.e,. c(w,q)
termPro[i]	the term probability of term i in the collection, i.e., P(w|C)
docN	the total number of documents in the collection, i.e., N
docLength	the document length, i.e,. |d|
docLengthAvg	the average document length in the collection, i.e,. avdl
termN	the total number of unique terms in the collection
MaxDF	the maximum document frequency in the collection
TFC[i]	the total number of occurrences of term i in the collection

3. Step-by-step guide

- Retrieval function
  - ✓ Create function
    - ◆ You can create a function with a single parameter setting or multiple

    ones as described in the previous section.
    - Two default functions are provided to help you started.
    - You can always start with opening an existing retrieval function, modify it and then save the modification as a new function.
    - Once you click the “save” button, the code will be checked for syntax errors.
    - You have to specify a name that has not been used in the system when you save the function, no matter it is a new implementation, or a modification for the existing functions.
  - ✓ Manage function
    - If a function has a single parameter setting, this function can then be used to create a search engine.
    - *If you want to create a search engine from a retrieval function with multiple parameter settings, please first start with this function, choose one parameter setting, save the revised function as a new one, and then create a search engine based on the revised function.*
    - You can always edit or delete a retrieval function that you have created.
    - You have to save the retrieval function with a new name after you edited the retrieval function.
    - You can also evaluate the retrieval function over different collection. (Just click the evaluation button! But be aware that you might need to wait for a few minutes for the results when the collection is large.)

- Search engines
  - ✓ Create search engine
    - You can create a search engine by selecting a retrieval function and a collection (i.e., index). You can also set up the privacy level for the search engine.
    - *Again, you can only select retrieval functions with a single parameter setting.*
    - Your search engine would allow you see the retrieval performance of your search engine and the relevance judgments for each result if you use one of the official TREC queries as your query (click “Official Query” to select the queries).
  - ✓ Manage search engine
    - You can delete your search engines
    - You can also follow the link to go to the search engines that you have created.
  - ✓ Compare search engines
    - You can conduct side-by-side comparison for two retrieval functions (with single parameter setting) that you have created. You need to select
      - Two retrieval functions
      - A test collection

            - A query (either enter your own query or select a TREC query)
        - The changes of document rankings are highlighted.

- Leaderboard
    - Summary
        - We summarize the most effective retrieval function for each collection.
            - The user can see how the function is implemented
            - Compare your functions with the best one
                - Per-query: side-by-side comparison of search results for a query
                - Overview: summary of per-query performance between two methods
    - Per-collection
        - We display the top 10 effective retrieval functions for each collection.
        - The users can see how each function is implemented.