
Information Retrieval Course

CSE 435/535 - Fall 2010

Project 3

Document Retriever in C++

Project Description In this project we will focus on retrieval of documents, extracting relevant information from Wikipedia meta files and computing quality scores of pages using PageRank. We will also have a competition component in this project.

Due Date 1: **Online submission through UBLearns' Digital Dropbox by 22nd November, 2010 11:59 P.M.**

Due Date 2: **Online submission through UBLearns' Digital Dropbox by 3rd December, 2010 11:59 P.M.**

1 Basic ranking and retrieval of News and Wikipedia documents

[40 points] due by 22nd November, 2010 11:59 P.M.

In this **first component**, you will be reading a query file in TREC format and retrieving and ranking documents.

Utilize the forward and the inverted indices built in Project 2 in order to implement the vector space model (VSM) as described in the textbook. Creating the indices correctly will fetch you **10 points** in this component.

The first task is to implement and compare the ranks of the documents retrieved using the following similarity measures (weighting schemes):

- **Boolean** - Boolean Model [10 points]
- **Cosine Similarity** - Standard Vector Space Model [10 points]
- **Okapi Weighting** - Probabilistic Model [10 points]

A sample TREC query file will be supplied and all the queries should be processed in a single batch. You will need to output your results in the format that is understandable by the “ireval” program that was demoed during the Wumpus search engine presentation. Look at files “topics.TREC” and “sample_output.txt” in the “project resources” section for Project 3 in UBLearns. For executing your program please follow the command line parameters from Section 5 in this project specification.

The format in which you have to store the output (so that the scorer ireval can compute the precision/recall values) is explained in the “**readme_eval.txt**” file. All columns in the “sample_output.txt” file are tab separated.

TO SUBMIT:

A pdf report containing 4 tables - 2 for one value of $K = 20$ and 2 for another value of $K = 100$. One of the 2 tables for each value of K contains MAP score and the other one contains

R-prec scores for “all” queries as given towards the end of the ireval output file. Each table contains 2 rows and 2 columns and the 4 tables are given below:

K = 20		
<i>MAP</i>	Your Team	Wumpus
Cosine		
Okapi		

K = 20		
<i>R-Prec</i>	Your Team	Wumpus
Cosine		
Okapi		

K = 100		
<i>MAP</i>	Your Team	Wumpus
Cosine		
Okapi		

K = 100		
<i>R-Prec</i>	Your Team	Wumpus
Cosine		
Okapi		

Our IR Model
<i>Brief description of your model</i>

2 Assigning global scores to documents using link analysis

 [40 points] due by 3rd December, 2010 11:59 P.M.

2.1 Computing PageRank

In this **second component** of the project, create a link matrix using the links extracted from the wiki documents only. Furthermore since your Wiki dataset is a random collection of 50,000 Wikipedia documents, some of the links will point to pages that are not in the collection. Those links can be ignored. Since the collection consists of internal Wikipedia documents, figuring out whether an internal Wikipedia link points to or out of an existing page is easy - you just need to compare the link string to the article name string. Create a link matrix out of those links and obtain (possible) global scores of the Wikipedia documents using PageRank. **You should also store whether a global score has been obtained for a document in a suitable index.**

2.2 Query focused intelligent document retrieval and ranking using interactive *live* search

This **second component** of project 3 also deals with modifying your code for the first component from section 1 so that it can accept queries directly from the keyboard (online), instead of reading the queries in a batch mode (offline). You are required to create an infinite loop for input processing which outputs “enter query: ” at each prompt. After search the program outputs “*Retrieved <so-many> documents in <so-many> seconds*” and lists the top 15 (or less) ranked **items**. The program will keep accepting user input until it encounters a blank line i.e. the user hits the enter key.

The nature of the items returned:

- If the query matches any Wikipedia document in the first 10 results, then the first item in the ranked list should be **information extracted from the Semwiki documents** and not an item that contains a link to the document. It should also mention the name of the Wikipedia article as source. The rest of the documents will be the filenames of the documents with each filename followed by a snippet. A

snippet is the shortest span in the text that contains most of the query words. If such a span exceeds 30 words, truncate it to 30 words.

- Show the PageRank by the side of each document filename followed by an explanation. The explanation is a string that says “[# of outgoing links in index from this page, # of incoming links in index into this page]”. Note here that a value of [0,0] implies that the page is a singleton and exists as a disconnected node in the document connectivity graph.

Note: A query file named “topics.training” will be uploaded to UBLearns. This query file will be in the same format as the TREC query file, however, you can just copy paste the queries from this file to the console and be able to search interactively.

3 Competition

[20 points] due by 3rd December, 2010 11:59 P.M.

In the **final (i.e. third) competition component** of Project 3, the task will be to try out interpreting the query. Each query in the “topics.training” and the “topics.TREC” file contains additional information which you can make use of for this task. In particular, each query has at least two fields a title and a narrative (and sometimes a description). The query to your system could thus be composed of only the string in the “title” field or the “description” field or the “narrative” field or a combination of the fields. Note that the description field will be containing full sentences highlighting information need and not just a bag of words like the one from the title string.

You will need to cleverly formulate a bag of words query from the different fields that reflect the information need as closely as possible. Your system will then be evaluated using the “topics.TREC” query file and the ireval program.

Furthermore, use your proprietary indexing *which can include champion lists*, **innovative ranking function**, better query interpretation module and anything else to get the best results on the evaluation amongst other teams. The results will be **quantitatively and qualitatively** evaluated using the output for the test query set as well as the time it takes to retrieve the results. *The output in this component **may** need to be connected to the browser using PHP - this is not decided at this time.*

The 5 best teams will receive standing ovation in class and possibly some gifts.

4 Data

The entire training data is available at /projects/TREC/CSE535_Fall2010 folder on CSE machines.

5 Strict submission guidelines

- As in project 2, you are constrained to create not more than 100 barrels. The command line interface to your indexing component will remain the same as in project 2.
- The command line interface to your program for the first component and the competition components are as follows:
\$ > dodo [-r | --retrieve] <path-to-index_store_dir> <path-to-dict_store_dir>
<path-to-query_file> [--boolean | --cosine | --okapi | --your-model] <zone>
<top-K> [-snp | --snippet]
→ The “[-r | --retrieve]” is the switch that says the program should be in the retrieval mode using a query file. Note that the switch string is either “-r” or

“--retrieve” (lowercase) using standard unix command line conventions. Top_K is an integer denoting the number of top ranked items retrieved; a value of -1 would mean “retrieve all items”. Note: <path-to-query_file> is the path to the query file. → The “[snip | --snippet]” indicates output of snippet in the resultset. If the command for retrieval uses this switch, snippets should be displayed for the resultset of **each** query in the query file.

→ <zone> is a string and can be “title,” “desc,” “narr,” “title+desc,” “title+narr” or “title+desc+narr”

- The command line interface to your program for the second and third components are as follows:
\$ > dodo [-s | --search] <path-to-index_store_dir> <path-to-dict_store_dir>
[--boolean | --cosine | --okapi | --your-model]
→ Snippet generation is default in this mode.
- The **name of your tarball submission** in Digital Dropbox should be <your_team_name_Project3-final> while the **actual filename** should be <your_team_name_Project3-final>.[tar | zip]

NOTE: If your program fails to run, the only way we can grade your project partially is by looking at your project documentation

5.1 Strict submission checklist

- Must submit makefile (0 points for not being able to run make)
- Create a folder named “resources” and move all data resources like “stopwords.txt” etc. to that folder
- Create a folder named “config” and move all configuration files like “inverted_index_config.txt” to that folder
- Make sure that the name of the executable is “dodo”
- Indexing as in Project 2 [10 points] [*Due by 22nd November*]
- Component 1 [30 points] [*Due by 22nd November*]
- Component 2 [40 points] [*Due by 03rd December*]
- Component 3 [20 points] [*Due by 03rd December*]

6 Submission

Submit a tar or zip file named <your_team_name_Project3-final>.tar or <your_team_name_Project3-final>.zip using the Digital Dropbox in UBLearnS. The contents of the compressed file is as follows:

```
-- <your_team_name_Project3-final>.tar  
|+ All source and make files  
|+ A pdf generated from Doxygen (required for project3)  
|+ A README text file similar to the one submitted for the first project. It should include  
the instructions to compile your code, the input arguments to be provided to your main  
program, etc.
```

Standard Message

Make sure all your programs at least compile and run on at least one CSE machine: (e.g. metallica.cse.buffalo.edu [running Linux]) For evaluation, the TAs will download your most recent tar files (upto the submission date) with the name <your_team_name_Project3-final>.tar from UBLearnS and run it. NOTE: There might not be a demo for project 3.

DO NOT TAR THE INDEX FILES AND SUBMIT THEM ALONG WITH THE SOURCE FILES. UBLEARNS CANNOT ACCEPT LARGE FILES. ALSO DO NOT EMAIL US YOUR TAR BALLED PROJECT - THEY WILL

**BE DELETED. WE ARE VERY STRICT ABOUT DEADLINES THIS TIME
- LATE SUBMISSION BY 24 HOURS DEDUCTS 10 POINTS, BY TWO
DAYS DEDUCTS 25 POINTS, BY THREE DAYS DEDUCTS 50 POINTS
AND DONOT SUBMIT AFTER 3 DAYS**