COMP4650/6490 Document Analysis Assignment 1 – IR

In this assignment, your task is to index a document collection into an inverted index, and then measure search performance based on predefined queries.

A new document collection containing more than 10,000 government site descriptions, and a set of predefined queries, is provided for this assignment. The provided code implements most of an information retrieval system. This provided code is designed to be simple to understand and modify, it is not particularly efficient nor scalable. When developing a real-world IR system you would be better off using high performance software such as Apache Lucene.

Throughout this assignment:

1. You will develop a better understanding of indexing, including the tokenizer, parser, and normaliser components, and how to improve the search performance given a predefined evaluation metric,
2. You will develop a better understanding of search algorithms, and how to obtain better search results, and
3. You will find the best way to combine an indexer and search algorithm to maximise your performance.

Throughout this assignment you will make changes to the provided code to improve the information retrieval system. In addition, you will produce an answers file with your responses to each question. Your answers file must be a .pdf file named u1234567.pdf where u1234567 is your Uni ID. You should submit both your modified code files and answer.pdf inside a single zip file.

Your answers to coding questions will be marked based on the quality of your code (is it efficient, is it readable, is it extendable, is it correct).

Your answers to discussion questions will be marked based on how convincing your explanations are (are they sufficiently detailed, are they well-reasoned, are they backed by appropriate evidence, are they clear).

Question 1 – Running queries (20%)

You have been provided with a simple implementation of an inverted index search system, which can be found in files *inverted_index.py, preprocessor.py* and *similarity_measures.py*. Your first task is to complete the *run_queries.py* script. The script should

1. Create an inverted index from gov/documents then
2. Run all of the queries from gov/topics and
3. Write the returned results for all of the queries to runs/retrieved.txt **in TREC_EVAL format.**

Step 1. has already been implemented for you. You will need to edit *inverted_index.py* to complete the run_query function so that it returns the highest scoring documents. You will then need to complete the *run_queryies.py* script to implement steps 2. and 3.

The Text Retrieval Conference (TREC) is a popular conference for research on designing better information retrieval systems. We will use their evaluation tool TREC_EVAL to measure the performance of your system. The tool expects the returned results to be in a file with a very specific format, each line in this file must be of the form:

query_id Q0 document_id rank score MY_IR_SYSTEM

Where query_id identifies the query that this document was returned for. Q0 is the literal string Q0. document_id is the name of the document that was returned. Score is the similarity score between query and document. Rank is the order this document was retrieved in (e.g. a rank of 0 means that this document was retrieved first, rank of 1 means second and so on). MY_IR_SYSTEM is the literal string MY_IR_SYSTEM. Once you have successfully created the file you should be able to run the *evaluate.py* file to see how well your retrieved documents matched the ground truth relevant documents for the given queries.

In your answers file to this question put a list of your system's scores for each of the evaluation measures, e.g

map: 0.1

Rprec: 0.1

recip_rank: 0.1

P_5: 0.1

P_10: 0.1

P_15: 0.1

Question 2 – TF-IDF Similarity (20%)

Currently, the inverted index uses TF similarity to calculate scores. Your next task is to implement TF-IDF similarity functionality. You will need to complete the methods of the TFIDF_Similarity class in the *similarity_measures.py* file. You are encouraged to study the provided class TF_Similarity to see how data can be extracted from the inverted index. There are many ways of implementing similarity using TF-IDF; while the lectures suggest weighting both the query and documents using TF-IDF this is not always the best choice in practice. Here you will implement the scheme described in SMART notation as *lnc.ltc* . The details of which are described in one of the course texts `*Introduction to Information Retrieval*, Manning C, Raghavan P, Schütze H, Draft April 1 2009` in section 6.4.3 and figure 6.15. The following is a reproduction of this figure (for more details see the textbook):

Term frequency		Document frequency		Normalization	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1+\log(\mathrm{tf}_{t,d})$	t (idf)	$\log \frac{N}{\mathrm{df}_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2+w_2^2+\ldots+w_M^2}}$
a (augmented)	$0.5+\frac{0.5\times\mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	p (prob idf)	$\max\{0,\log\frac{N-\mathrm{df}_t}{\mathrm{df}_t}\}$	u (pivoted unique)	$1/u$ (Section 6.4.4)
b (boolean)	$\begin{cases}1 & \text{if } \mathrm{tf}_{t,d}>0\\ 0 & \text{otherwise}\end{cases}$			b (byte size)	$1/CharLength^{\alpha}$, $\alpha<1$
L (log ave)	$\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t\in d}(\mathrm{tf}_{t,d}))}$				

► **Figure 6.15** SMART notation for tf-idf variants. Here *CharLength* is the number of characters in the document.

The mnemonic *lnc.ltc* corresponds to a weighting scheme of *lnc* for the document terms and *ltc* for the query terms. The first letter of each scheme corresponds to the term frequency weighting, the second corresponds to the document frequency, and the last corresponds to the normalization.

After you have successfully implemented TF-IDF Similarity as described you can change the *sim* variable in *run_queries.py* to TFIDF_Similarity. Next, Run *evaluate.py* with your new TFIDF_Similarity index. Write down your new evaluation scores.

Question 3 – Evaluation measures (20%)

Did your system perform better with TF_IDF or TF similarity? You will need to decide which of the provided evaluation measures is most useful for evaluating this system. In your answer you should state which measure(s) you are basing your decision on, and why your chosen measure(s) are appropriate specifically for this gov corpus. If you need to make any assumptions about how the system will be used, then make sure to state them.

You should continue using the best performing similarity measure for the next questions.

Question 4 – Improving the pre-processor (20%)

The pre-processor defines how raw text is converted into tokens to be indexed. Currently, it does whitespace tokenization and porter-stemming. Your task is to change the Preprocessor class to improve your system's evaluation. You may want to try changing the tokenizer, the stemmer, or any implement any other technique mentioned in the lectures (eg normalization, stop word removal). You are strongly encouraged to make use of the NLTK library for this task, documentation can be found here: https://www.nltk.org/api/nltk.html.

Briefly describe **four** different settings of Preprocessor that you tried, and what impact they had on your system's performance. Explain *why* you think that each change increased/decreased your model's evaluation measures. Write down your evaluation measures for the best performing settings that you found.

Tip: Make sure you change the *used_stored_index* flag to False in *run_queries.py* to force your index to be rebuilt with your new pre-processing.

Question 5 – Further Modification (20%)

Your final task is to implement some further modification to the system in order to improve its performance. Exactly what kind of modification you implement is left up to you, some examples of modifications you could make:

- Implement BM25 similarity for the system (http://ipl.cs.aueb.gr/stougiannis/bm25.html).
- Adjust the inverted_index to use different weighted fields.
- Implement some domain specific pre-processing.
- Implement a different document and query weighting scheme.
- Anything else you can think of.

In your answers describe your best Information Retrieval system, write down its evaluation measures, and explain why you think your changes improved the measures.

Academic Misconduct Policy: All submitted written work and code must be your own (except for any provided starter code, of course) – submitting work other than your own will lead to both a failure on the assignment and a referral of the case to the ANU academic misconduct review procedures: ANU Academic Misconduct Procedures