Programming Assignment 2 - Retrieval Models Report

Submitted By: Ankita Mehta

October 20 , 2017

1 Retrieval API

Solution:

1.1 API Retrieval Design:

After creating compressed, uncompressed indices (in the previous assignment) , next task is to generate the document scores using Document_at_A_time_Retrieval method. It will give you the score for every document in which at least one of the query word is occurring. This code is using only the uncompressed indexes. This wrapper takes into account three command line arguments:

1. ***-q or −−query_path*** : It is the mandatory argument, which specifies path to query file.
2. ***-m or −−manifest_file_path*** : It is also a mandatory argument, which specifies path to manifest file generated while creating indexes. This file will define whether Retrieval has to be done for compressed or uncompressed file.
3. ***-no or −−model_No***: It specifies which retrieval model is to be used: 1 for BM25 , 2 for QL-JM , 3 for QL-DIR , 4 for VSM.

1.2 Design Tradeoffs:

1. In all the models, a copy of Inverted lists of query words have been made for calculating the doc score. This has been done because, rather than taking pointers for every list and maintaining them, the temp set of inverted lists has been updated on the go.
2. Also, the document scores have been stored as negative numbers. Because in Python, in priority queue the element with minimum value comes first. So, to retrieve the document with maximum score at first, document scores have to be stored as negative integers.
3. For BM25, tf and qtf are taken as the raw scores.
4. While implementing VSM model, document and query vectors have been made as the sparse vectors. This has been done because in python, numpy computation is pretty fast and the vectors can be used for the further projects as well.
5. Also, for VSM, tf is taken as (1+log10(tf)) , idf : log10(N/n_i) and weight of document is taken as tf*idf. Then every document vector and the query vector is normalised before finding the dot product between them.
6. All the logs are taken as log10.

1.3 Various Questions and how they were approached:

Various difficulties that were faced while doing the project and how they were approached:

1. Whether the query considered has all the unique terms or all the terms.
2. The log of tf, idf should be taken or not while implementing BM25/VSM/QL models. How the log of values are affecting the scores. I figured out these problems after having discussion with the professor.

2 Do you expect the results for Q6 to be good or bad? Why?

Solution:
According to me, the results of Q6 should come bad. Because we are finding out the score of a document considering the queries as the bag of words not the phrase i.e, we have bunch of words in a query and then in whatever document if atleast one word is present, we are giving some score to that document for a given query. The query 6 (to be or not to be) , has all the common occurring words. These words would be present in almost every document, so the score that will come will not be that relevant. This query returned 744 documents out of 748 documents.
However, if we consider the document which has all the terms of the query present (in whatever sequence) as a good result, then QL models performed well for Q6 as compared to BM25.

3 Do you expect the results for a new query "setting the scene" to be good or bad? Why?

Solution:
I expect the result of this query not as bad as the query "to be or not to be". Since it has atleast one word which is not the common occurring word.

Since we are considering the query as the bad of words, so the result of this query is totally dependent on the word "setting". This is because "the" and "scene" are present in every document. Word "setting" is occurring 25 times in whole 748 collections.
For BM25, it didn't give relevant results, because of the log(idf) term which decreases the score of the document. So, it is possible that one document having instances of "the" and "scene" could overpower the document having all three words , i.e it could be possible that the score of document having only "the" and "scene" is greater than the document having all three words. ($score(a + b) > score(a1 + b1 + c1)$).
Again if we consider the document which has all the terms of the query present (in whatever sequence) as a good result, QL models performed better in this as well.

4 Judgments.txt

Solution:

I assign the relevance 3 to those documents having all the query terms, 2 if one is missing and so on.
13 Unique scenes have been found in 3 models:

1. antony_and_cleopatra:4.1 3
2. coriolanus:0.6 2
3. antony_and_cleopatra:3.13 2
4. timon_of_athens:2.2 2
5. coriolanus:0.9 2
6. antony_and_cleopatra:3.8 2
7. henry_vi_part_3:3.1 2
8. henry_vi_part_3:3.2 2
9. antony_and_cleopatra:3.2 2
10. julius_caesar:3.1 2
11. henry_vi_part_1:1.0 2
12. henry_v:3.3 2
13. antony_and_cleopatra:3.5 2

5 What will have to change in your implementation to support phrase queries or other structured query operators?

Solution:
Various things need to be changed are :

1. Currently, scores are calculated on the basis of if atleast one term of the query is present in the document. To implement the phrase queries, I would need to check if the whole phrase taken as a single entity is present in the document.
2. We could also maintain the counts of all n-gram phrases for running the phrase queries.
3. We have already implemented dice coefifcient, which could be modified a bit for considering ordered and unordered query operator separately.

6 How does your system do? Which method appears to be better? On which queries? Justify your answer.

Solution:
All the models performed equally well except some of the instances. For example, for queries like Q6 (to be or not to be) , BM25 model didn't perform very well. This happened because the 1st term in BM25 implementation is log(idf) and because of common occurring words , this term will decrease the score of the document. So, it is possible that one document having instances of one word of query could overpower the document having all words of query. QL models seems to be performed better for all types of query, because it calculates the probability of a query given the document and it also gives some score to a document even if a word is not present in that. Being a generative model, this performed better than rest all models for a small dataset. But could perform very bad if the size of dataset increases. Also,VSM model performed bit better than BM25 for Q6.