Programming Assignment 5 - Query Independent Features

Submitted By: Ankita Mehta

December 07 , 2017

1 Query Independent Features

1.1 Implementing prior feature

After creating compressed, uncompressed indices , Retrieval models and the inference network (in the last assignments) task of this assignment is to test the belief And operator with the prior feature on the shakespeare dataset. This has been implemented in the following steps:

1. Firstly indexes and vocabulary was created and then various structured query operators and belief node operators were implemented as a Dynamic Dispatch (as a part of previous assignments).
2. Then as a part of this assignment, functionality for 'prior' feature has been added in the code where the belief node is computed.
3. To test the belief_AND operator with 'prior' feature and the query 'the king queen royalty' , a structure has been created manually, which can be seen in the code: 'test_belief_with_prior.py'

1.2 Design Tradeoffs:

1. random.prior and uniform.prior has been dumped as a dictionary, because for the belief node computation, a list of dictionaries having documents and their scores have been passed. This implementation has been done by keeping in mind that the dataset is small.
2. In belief operator, rather than passing a single document, a list of document-score dictionary has been passed.
3. The belief nodes and its children should be passed in the specific format. It should be a structure/class with data members as : type (denotes the type of node : a belief operator / ordered window /unordered window / term(s)) , children: which is of same structure as the query node and these can be a belief operator / ordered window /unordered window / term(s), value : which will store the name of belief operator or the query. This design has been chosen as compared to the tree format because of it is easy to understand and implement.

1.3 Various Questions and how they were approached:

Various difficulties that were faced while doing the project and I figured out these problems after having discussion with the professor and peers

1. What values should be present in uniform.prior like 0 or 1 or could be anything of our choice ?
2. Does the probabilities assigned should be between 0 and 1 ?
3. Do we have to save the results only for one query ?

2 What is the difference between your two query runs? Why would it be that way? Be specific.

Solution:
The prior probabilities influence the rankings by preferring documents with certain characteristics. We have run the query for two priors : uniform and random. Uniform prior feature gave equal weightage

to all the documents. So even with the multiple runs, it will always give the same results. Whereas the random prior assigns the random and different probabilities on every run, so the ranking would also change with every run.
But the fun fact here is that, the 1st ranked document always remained the same even with the multiple runs in random prior features which is : romeo_and_juliet:2.3. This has happened, because there is no query(the king queen royalty) word present in this document and rest all documents have negative scores. So given that random probability is in between 0 and 1, it was given the highest score, hence the 1st rank. Because of the same reason, the 1st ranked document is the same with uniform prior probability feature as well.

3 How should the priors be stored in the index? Raw probabilities? Log probabilities? Some other value? What should drive your choice? Be specific

Solution:
Using the prior probabilities, once can influence the rankings of the documents. Priors can be stored in any way, either raw or log because Multiplication of raw probabilities is same as the summation of log probabilities, so in short their storage won't affect the raking of the documents. Taking the log of raw probabilities just map the higher values to the lower values without changing the relativeness amongst them.