COMPSCI 546: Assignment 4

ppachpute@umass.edu

November 2019

1 Questions

1. Description of the system, design trade offs, questions you had and how you resolved them, etc. List the software libraries you used, and for what purpose.

   - Now the system is able to evaluate a query using the inference network. It does evidence combination to calculate the score of a document.
   - API functions like skipTo(), nextCandidate() etc. were not previously written for posting list. Now that inference network uses those functions heavily hence I added those API calls to the existing Posting List implementation.
   - Previously I used to return the same Posting List every time that was requested to Inverted Index. Now, I have added a new function that returns the copy of a Posting List. This is needed for queries like "to be or not to be" where the two "to"s require different Posting List.
   - In Term Node and Window Nodes, apart from generating the Posting List all the functions are encapsulated in Proximity Node thereby there are no code duplicates.
   - I have also created a class WindowNode which sits between Proximity Node and Ordered/Unordered Windows. This is an abstract class that implements many of the functions (like finding the document that contains all the terms etc.) needed by both the window nodes.

2. Look at the top ten results for Q6-Q10 using the output from the ordered and unordered window operators. Using the same relevance judgment process as the previous retrieval models assignment, evaluate the difference between the two window operators. Explain any differences in behavior. Anecdotal evidence is sufficient.

   - The two operators serve different purpose in query evaluation. Ordered window operator is designed to be more in lined to evaluate phrasal queries effectively while Unordered window operator is designed to be more in lined to evaluate topical queries effectively

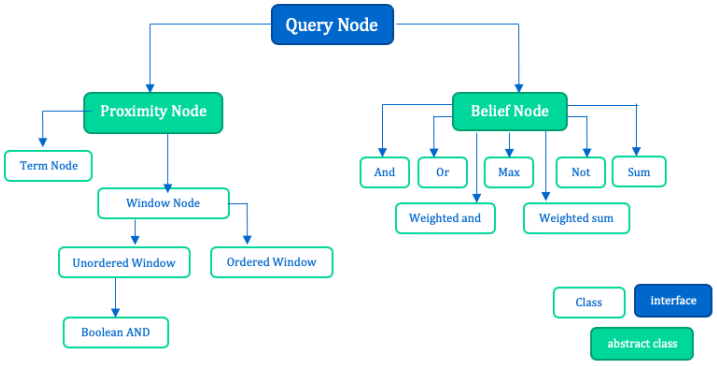

Figure 1: Class structure

- The above claim was justified by the query "to be or not to be". The query seems more like a phrasal query than a topical hence the result should contain only the documents in which the phrase is present which is exactly what the Ordered window operator has done while Unordered window operator has returned more documents which do not contain that query.
- Similar results can be seen for query 8 which is "alas poor" It also seems a phrasal query hence Ordered window has returned documents where it was a exact match while Unordered window returned documents in which the query terms "alas poor" appeared in proximity but the context was not the same. Snippet from one of the play which Unordered window returned for query 8: "execution isabella alas what poor ability s in me to do him good lucio assay". I think this not what user was expecting.
- However, if we consider some other example like "bat ball" then the query seems topical i.e. the user seems interested in the documents related to cricket while it may not be important to have those two words in the exact same order. In such cases Unordered window operator would work better.

3. What needs to be done to implement a structured query language for your retrieval system. Sketch out the design.

   - Important thing to implement structured query language is to have a query parser and inference network generator. That is, given a query how to formulate the query nodes and how to add dependencies between them.
   - To implement SQL we will first have to write a query parser that would tokenize the input string into individual words. Post that individual query nodes would have to be generated.

- Now, Based on the way nodes were organized in the input query we would have to handle creating the respective children. For example if the query is regarding abraham lincoln or president lincoln then input query would resemble to something like syn(od:1(president lincoln) od:1(abraham lincoln)) where "abraham" and "lincoln" are terms inside of ordered window. The two ordered window are children of syn operator.
- Moreover, precedence would have to be defined as well. Like the innermost nodes are created first and then those are assigned to enclosing node as children.
- Filter nodes would have to handled cleverly like propagating them up the network so that those are evaluated at last and correspondingly results are returned.

4. How do you expect each of the combination functions to behave, both individually, and in concert? What impact would normalization have on the and and weighted and operators?

    - Depending on the type of combination function the expected outcome could be different.
    - Like in case of AND operator it would rank those documents higher in which all of the or most of the query terms are present. In case of weighted and it would favour the documents which have high score as well as have high weight.
    - OR function would favour the documents in which the occurrence of term or combination of the terms is maximum.
    - In Concert like AND (OR, OR), first inner OR would provide the results which would favour the documents having high term frequency / occurrences of words. Once such documents are obtained the AND operator would try to find the documents which contain both the OR terms and would favour such documents.
    - In case of Normalization, if the normalization is done on just the just a single query node and that node is not used in any combination then the resulting ranked list would be unchanged as the scores would change but the relative ordering would be preserved.
    - Same would be the case if the normalization is done with another and operator like AND(NORM_AND AND). Even in this case the resultant list would be same as the one with no normalization.
    - If the normalized and is used in conjunction with operator like max then it would make a difference. Consider query MAX(AND, OR) and MAX(NORM_AND, OR) in the latter's case score would be lower (if the scores are positive, and higher if the scores are negative) than the initial query and hence the results would change.