# COMPSCI546 Assignment-4 Report

Jaideep Rao

October 2020

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

The system implemented in this project employs a hierarchy of different types of nodes that are semantically related to each other. All nodes implement the Query node interface and are further categorized either as a filter node, a belief node or as a proximity node. Each of these mentioned categories are implemented as abstract classes that contain implementations of behaviour that would be common across all nodes that inherit them, and also allow for subclasses to define their own use-case specific implementations for some methods. Every non-leaf node is defined as abstract class that extends a parent abstract class and is extended by its children. At execution time we create query trees by hand for a specific set of queries and operators we want to apply to those queries. Once the query graph/tree is appropriately constructed, it is passed into the inference network to be executed and to produce the list of ranked documents for each query. In terms of some specific design choices made in this project:
1. I chose not to double dip during window creation, by advancing the position of each pointer so that we don;t consume the same character in multiple windows and each character only contributes once
2. I also chose to main a separate array of pointers into each of the position arrays so that we can store the last visited position for a particular array and start traversing from there and avoid scanning the loop from the beginning each time unnecessarily
I used the standard java.lang and java.util libraries

## 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.

Looking at the results produced by both window operators, they behave very similarly in some cases but they diverge in some other cases. Neither outputs anything for Q10 since the phrase ”antony strumpet” doesn’t occur anywhere. Similarly, for Q7, since it is a single word query, they effectively output the same thing. The same applies for Q9 as well, in these cases both the operators output identical results in terms of ordering and scores. The most significant deviation can be seen in Q6, where the ordered window directly outputs the exact phrase which only occurs once in the entire collection, whereas the unordered window outputs a list of results due to having a wider scope of operation and as a result may not output results as relevant to the search

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

In order to support a structured query language for this system a significant number of things would need to change or be added to this system. For example, we would need to define the structure and rules for how queries can be constructed for this particular system. We would need to be able to parse a provided structured query (potentially of the form uw4(od1(white, house) and od1(washington,DC) or USA) and be able to correcty identify the operators involved and their operands. On having identified these things, we would need to implement a query-tree builder that can appropriately understand the hierarchy and relations between these objects to accurately build them into the tree (as opposed to doing it manually). Further, in order to facilitate any potential nested windowing operations, we would need to modify our existing logic to store extents for each window (start and end positions) rather than just storing the starting positions for each encountered window.

In terms of changes in the design of the system, we would have to add a dedicated class to construct the query tree, a call to which could replace the manual tree-building currently in our evaluation programs.We might also need a dedicated query-extractor that can parse the provided query to identify the relevant operators and their parameters as well as discern which operators act on specific terms and sub-queries

## 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?

When combination functions are applied individually, it would be fairly straightforward to determine how these operators would behave since we know the exact computations or at least relative scores (to each other) that would be generated for a set of documents based on the operator being used. However, it would become harder to predict how a combination of such operators would behave since the output might vary significantly with changes in the structure/hierarchy of the query tree that implements these operators
The impact of normalization on the and and weighted and operators would be to bring them closer in behaviour to the sum and weighted sum operators (since they are also normalized). It would allow us to describe the belief computation of these operators in terms of various types of means (averages). Therefore, a weighted and would compute the weighted geometric mean whereas the normalized weighted and would compute the geometric mean (ref pg 278 of textbook )