Report: Ranking on query results

Lukas Donkers - 5523265
Roel van Warmerdam - 4300556

Making the Metadatabase

Reading the workload

The workload is read per line (after the first 2) as each line does the same, these lines are splitted in two groups first, one is the amount of times that query is entered, two is the query itself. This query is stripped of the SELECT …. FROM autompg WHERE since these values have no use in any of the other calculations. Then the remainder is splitted on the AND values to get all of the tables separated. These values are used in the QF and Attribute Similarity.

QF & Attribute Similarity

QF is calculated in two ways by checking if the value from the work contains IN or not. If it does it means that the corresponding table has multiple values to use for that table and will add the amount of times the query is entered to the RQF of that value, also the attribute will be used for the attribute similarity. If not there is only one value for that table in the query then only that value RQF is added. Then if all the queries are used the QF is calculated using QF(q) = (RQF(q)+1)/(RQFMAX$_{column}$+1), this prevents a value having a QF of 0 or worse dividing by zero. Attribute Similarity is calculated at the same time as QF, as it uses the same data from the workload. For Attribute Similarity, we only save the queries a value occurs in. For example, opel and saab occur in queries 44 and 68 (line 46 and 70 in workload.txt). The Attribute Similarity is a string that saves all these occurrences. For opel and saab the Attribute Similarity value would look something like this: '31 q44,13 q68,'. This allows us to easily find overlapping occurrences later.

IDF

The IDF is calculated in two ways, one for the numeric values and one for the others. The first method takes each values from a column and calculated the bandwidth using the following formula h = 1.06σn$^{-⅕}$ and then stores it with the formula as seen on the right. The second method uses the standard IDF method as taught in class: log(N / df$_t$)

$$IDF\ (t) = \log\left(\frac{n}{\sum_{i}^{n} e^{-\frac{1}{2}\left(\frac{t_i - t}{h}\right)^2}}\right)$$

Finalizing the Metadatabase

After all QF, IDF and Attribute similarity values are calculated, it is time to place them in the metadatabase. We save all values in a large dictionary for each table. For each table, we insert all values in the dictionary into that table through SQL Insert into statements.

Handling CEQs

Connecting to the databases

If the databases exist, connection is done at the start at the application, by finding the sqlite file and creating a sqlite connection like in Tigran's blog example. If the databases do not exist, the application will automatically make them, and then connect to them.

In order to prepare for incoming CEQs, the CEQHandler will query all database (not metadatabase) document data and save it locally. This means that document scoring for any query makes use of the local copy, and does not generate more database queries.

Similarly, bandwidth data from the metadatabase is saved locally, but all other metadatabase data will only be queried if a CEQ requires it.

Scoring documents

The scoring of the documents goes as follows, for each of the requested tables we check whether that table is requested, and if so that will be used to calculate the score, else the score will be used for the missing values on which more will be explained later. If the table has numeric values the IDF will be calculated using the formula to the right, otherwise we just use the value from the Metabase.

$$S(t,q) = e^{-\frac{1}{2}\left(\frac{t-q}{h}\right)^2} IDF(q)$$

The QF is also received from the database and used, unless the tuple doesn't have that value and that value isn't numeric.

Attribute Similarities are saved in the metadatabase as strings of query identifiers, we must first calculate the Jacquard coefficient using two W-values - one from the query, and one from the document. We do this using the Jacquard function, which calculates the total amount of queries of their intersection and their union, and divides the two.

Lastly, the score for that column is calculated by S=J*QF*IDF and the sum of all column scores becomes the score of that document. However this may result in ties, in that case we just want the best from the ties and that is where the missing values come in. All of the QF values of the columns which are missing in the CEQ and used in: $\log(QF_i(t_i))$ and then summed. Since this will al be negative the document tied with the highest missing score will be used first and so on.

Sorting the documents on score

While documents are scored, their score is saved in a list of tuples. These tuples contain the id of the document, their score, and their missing attributes score.

This list of document tuples is then sorted based on score - if the score for two tuples is equal, they are sorted based on missing attributes score instead (for tie-breaking).

After sorting, a list of tuples (including document ID's) remain, of which we can take the top-k, and display the documents with that ID (using the local database copy).