MSc. Computer Science – Data Analytics Xue Yang 19230949

Q2 (a) We can use local analysis to automatically expand query without explicit user feedback based on the returned answer set.

Step 1: In local analysis, a term-term ~~matrix~~ correlation matrix $M_{i,j}$ is created to quantify the connection between term i and term j. There are many different means to develop the correlations which are shown below:

- Association Clusters — calculate the number of co-occurence of term i and term j in the returned document collection using: $M_{i,j} = \frac{freq_{i,j}}{freq_i + freq_j - freq_{i,j}}$
- Metric Clusters — improve accuracy by considering the positions of terms within a document.

$$M_{i,j} = \sum_{t_i, t_j \in D_i} \frac{1}{dis(t_i, t_j)}$$

Note that when term i and t_j are in different documents, $dis(t_i, t_j) = +\infty$

- Scalar clusters — improve accuracy by considering the context of the query based on the assumption that if two terms have similar neighborhoods there is a high correlation between them. In this approach, similarity can be calculated based on comparing the two vectors representing the neighborhoods using the cosine similarity measure. Use this to define a term-term correlation matrix and the procedure continues as before.

Step 2, then we can develop an association cluster for each term t_i given the i^{th} row from the matrix and select the top N values from the row. These are the values which correspond to the top N correlates for term t_i. After the cluster for each query term has been created, we can get $|q|$ clusters. Note that N is usually quite small to prevent the query from becoming too large and potentially 'drifting' in terms to topic of the query.

Step 3. Add these new terms to expand the query. Also, may take all terms, or those with the highest summed correlation and itemize.

Potential limitations:

① Only re-retrieval the returned answer set, thus, may lack diversity of new terms added,

② Also, the improvement of performance cannot be guaranteed if the answer set is too small. Instead, searching through the whole collection set may bring a higher recall and performance.

③ Term ambiguity may introduce irrelevant statistically correlated terms, leading to topic-drift problem.

(b) Prediction of query performance is so-called query difficulty. The basis is how well the topic of the users's query is covered by retrieved documents. Existing methods of predicting query performance can be ~~dived~~ divided into two categories: pre-retrieval methods and post-retrieval methods.

Pre-retrieval predictors utilize the static information of a query, which can be computed before retrieval including Linguistic approaches and statistical approaches. The former method need to use ~~any~~ NLP techniques to analyze the query, and also use external sources of information to identify ambiguity. The latter method—statistical approaches involve the following aspects:

① take into account of the distribution of the query term frequencies in the collection like idf and icf of terms.

② take into account specificity of terms and queries containing non-specific terms are considered difficult.

③ consider term relatedness and if query terms co-occur frequently in collection, we expect good performance.

④ consider query scope. For example, what percentage of

documents contain at least one query term. If a lot then this is probably a difficult query.

Besides the static information, post-retrieval predictors also utilize the dynamic information, which can only be computed after retrieval. For example, Looking for coherency and robustness of the retrieved documents. We can classify these methods into:

① clarity based methods which are to measure the coherency (clarity) of the result set and its separability from the whole collections of documents. The language of the result set should be distinct from the rest of the collection. We can compare language model induced from answer set and one induced from the corpus / collection. This is related to cluster hypothesis.

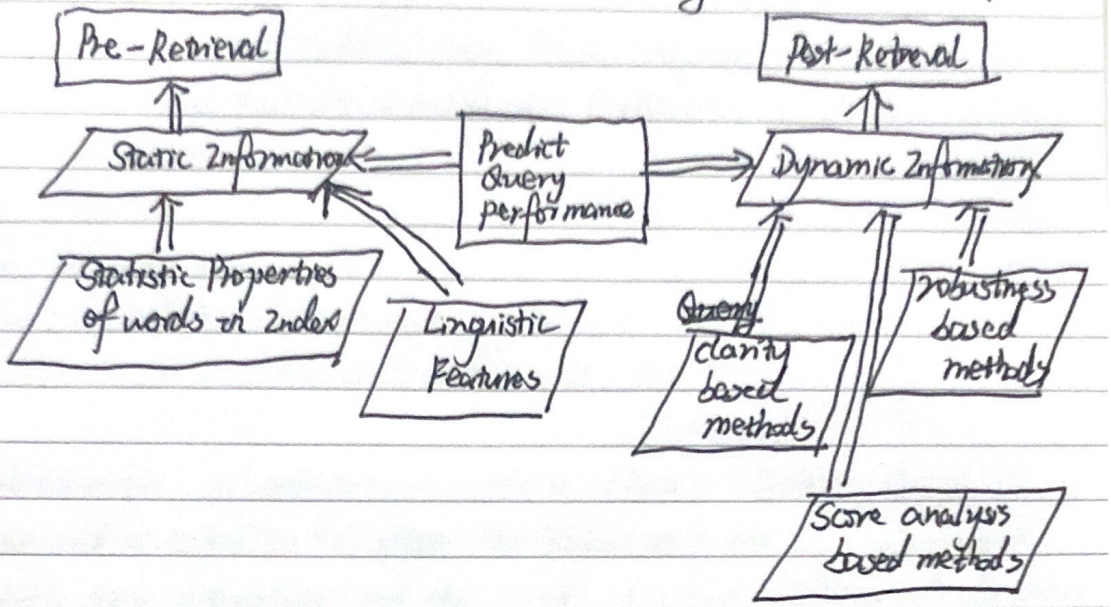
② robustness based methods that estimate the robustness of the result set under different types of perturbations.

- Query: overlap between query and sub-queries. In difficult queries some terms have little or no influence.
- Documents: Compare system performance against collection C and some modified version of C.
- Retrieval performance: Submit some query to many systems over some collection; divergence in results tells us something about difficulty of query.

③ score analysis based methods that analyze the score distribution of results where cluster hypothesis is supported. We can look at distribution of scores in answer set and document set and attempt to gauge difficulty. This is a relative simple measure which is shown to be effective.

Overall, we can predict which query is likely to perform better based on the above measures. And I draw a diagram ~~in the next~~ to better show these. (next page)

Xue Yang 19230949



(c) We can use experimental approach for testing the performance of the system.

① Firstly, a known collection of ratings by users over a range of items is decomposed into two disjoint subsets: The first set (usually the larger) is used to generate recommendations corresponding to those in the smaller set.

② Secondly, these recommendations are then compared to the actual ratings in the second subset.

③ Thirdly, the accuracy and coverage of a system can thus be ascertained where coverage measures the ability of the system to provide a recommendation and accuracy measures the correctness of the recommendations generated by the system. Besides, the accuracy is usually presented as the mean absolute error (MAE) between ratings and predictions.

Xue Yang 1923094g

Q3 (a) A good weighting scheme can highly improve the system performance. It comprises local factors, global factors (collection wide) and query related features:

$$sim(q,d) = \sum_{t \in q \cap d} (ntf(D) \times gw_t(C) \times qw_t(Q))$$

where ntf(D) is the normalised term frequency in a document; $gw_t(C)$ is the global weight of a term across a collection; $qw_t(Q)$ is the query weight of a term in Q.

For example, a well-known weighting scheme BM25/Okapi is created based on the above. It penalizes the long document, high term frequency and the words in a large number of documents, providing high retrieval accuracy.

The formula is:

$$BM25(Q,D) = \sum_{t \in Q \cap D} \left(\frac{tf_t D \times \log\left(\frac{N - df_t + 0.5}{df_t + 0.5}\right) \times tf_t Q}{tf_t D + K_1 \times (1-b) + b \times \frac{dl}{dl_{avg}})} \right)$$

where $\log\left(\frac{N - df_t + 0.5}{df_t + 0.5}\right)$ is the global factor; $tf_t Q$ is the query related factor; and $\frac{tf_t D}{tf_t D + K_1((1-b) + b \times \frac{dl}{dl_{avg}})}$ is the local factor which denotes the normalised term frequency in a document and also indicates the penalization of long documents.

(b) Axiomatic approaches have been developed which refine a number of constraints (axioms) to which all good weighting schemes should adhere.

① Constraint 1: adding a new query term to the document must always increase the score of the document.

② Constraint 2: adding a non-query term to a document must always decrease the score of the document.

③ Constraint 3: adding successive query terms to a document should increase the score of the document less with each successive addition. For example, encountering a term 30 times does not

increase the likelihood of relevance by the factor of thirty.

④ Constrain 4: ensuring that the document length factor is used in a sub-linear function will ensure that repeated appearances of non-query terms are weighted less.

The advantages of the axiomatic approaches include providing a means to identify where certain weighting schemes may be flawed and a means to guide the search for weighting schemes.

(c) Based on the scenario of the case, a recommender system that combines both content filtering and collaborative filtering can be used.

Firstly, when a user comes up with a new query, the terms in the query will be searched in the large repository (query log evidence). Then a term-term matrix will be generated. Select the history queries which contain the terms in the new query. It is obvious that the history query which contains all the terms of this user's new query is the one that has the highest similarity. Then the other terms appearing in the history query which are neighborhoods (i.e. bigrams or trigrams) of the target terms could be added to the new query. If the new query's terms are related to many history queries, we can also select the neighbords in each related history query to add to the new query.

Note that because the timestamp of every query is recorded, the system should consider the ~~term~~ time factor. For example, the newest history ~~quie~~ queries are much more useful than the ~~rel~~ relatively older queries as users' interests and opinions may change overtime. Thus, the system can make a threshold to indicate that the queries in which time period should be re-recommended. The threshold can be set as 5 years. Then the system can analyze the timestamp of the query to see whether they are within 5 years

Secondly, based on the search history of each user, a user-user matrix is generated by calculating Pearson correlation. So, the groups or neighborhoods of users who are similar are created. For example, a user who has used the same term(s) in a query can be matched. Then the system will search for the most correlated IDs with the user ID of the new query, and then recommend the top K terms appearing in the correlated IDs' history queries. This is because the users who are similar are likely to ask similar questions. Also, this is a kind of collaborative filtering. We should notice that there is a special situation. When the new query is from a new user, we do not know who is similar to him/her. Here, the system can use a weighted average of global mean and user's (or term's).

Finally, use a weighting scheme that involves user-user similarity score and query-query similarity score to distinguish which new terms should be added to the user's new query. According to this, the system can combine content filtering and collaborative filtering and return extra recommendations. Besides, we can even assign suitable weights to collaborative filtering CO and content filtering respectively. For example, at the beginning, these weights are equal. But when facing a new user or item, the weight of content filtering score can be increased, and the weight of collaborative filtering can be decreased.

Advantages: ① The system can give good prediction performance, especially a high recall rate because it combines the filtering of content and users' ratings. And the ignoration of duplicate terms ensures the diversity of predicted terms.

② The system can cluster similar users in advance and use these clusters to predict in a very efficient manner.

③ The system can suggest potential interests to users

7

especially to those who firstly use the system, and also recommend items that they were not expected.

④ The system consider timestamp which improves accuracy.

Disadvantage: ① the matrix is very large leading to data sparsity.

② ② It is hard for the system to distinguish synonyms. Especially, people of different ages or in different positions may like to use different terms to describe a query.

③ Because the order of terms is ignored, the weights of new terms cannot be correctly calculated, as well as the context may be mispredicted.

8

(CT4100)
MSc. Computer Science – Data Analytics Xue Yang 19230949

Q4. (a) The tf-idf weighting scheme is very popular and useful. However, in this case, we should modify tf-idf weighting scheme based on some specific attributes of the scientific articles. For example, the terms which appear in the title, keywords, abstract, authors, bibliographies have higher resolving power. We should involve these factors to tf-idf weighting scheme.

Let the weight of a term appearing in the main content of the document be $tf(c) * idf$; the weight of a term appearing in the authors be $\alpha tf(a) * idf$; the weight of a term occuring in the bibliographies be $\beta tf(b) * idf$. Then the weighting scheme function is shown below:

$$w_{ij} = f(tf(c) * idf, \alpha tf(a) * idf, \beta tf(b) * idf)$$
(where $\beta > \alpha > 1$)

Thus, the weight of a term i in an article j is:

$$w_{ij} = tf(c) * idf + \alpha tf(a) * idf + \beta tf(b) * idf$$

The vector of an article j can then be denoted as:

$$D_j = (w_{1j}, w_{2j}, w_{3j}, \cdots w_{nj})$$

The similarity of documents can be generated by calculating the cosine of the angle of their vectors. For example, the similarity of D_1 and D_2 is:

$$\text{Similarity}(D_1, D_2) = \frac{w_{11} w_{12} + w_{21} w_{22} + \cdots w_{n1} w_{n2}}{\sqrt{w_{11}^2 + w_{21}^2 + \cdots w_{n1}^2} \times \sqrt{w_{12}^2 + w_{22}^2 + \cdots w_{n2}^2}}$$

(b) Based on the above, the weight of a term k in the query is: $w_{kq} = tf(k,q) * idf$, thus, a query can be denoted as:

$$Q = (w_{1q}, w_{2q}, \cdots w_{nq})$$

The similarity of a document j and a query is $sim(Q_j, Q)$. The articles are then ranked descendingly by their scores and the top "p" articles are the answer set.

Given the above, the company can wish to rerank the articles based on how authoritative or influential the papers are. Here, HITs algorithm can be used.

First, the answer set in the above step is the root R. We initialize base set S to R. And add to S all the articles pointed to by any paper in R; Also, add to S all pages that point to any page in R.

Secondly, assign to each article $p \in S$:

an authority score: a_p (vector a)

a hub score: h_p (vector h)

And then initialize all $a_p = h_p$ to some constant value, say, let the constant be 1.

Here, the authority score denotes how authoritative or influential the article is. If an article appearing in a large number of good articles, then it normally has a high authority score. If an article points to lots of good authorities, it has a high hub score.

Thirdly, we define M to be the adjacency matrix and make $M_{ij} = 1$ where $i \to j$;

$M^T M$ can generate the authority vector a. And MM^T can return the hub vector h. Then, we can update the authority and hub scores following: $a \leftarrow M^T M a$

$h \leftarrow MM^T h$

The above updates will iterate until convergence and the final authority and hub scores are generated.

Now, the IR system of the company can re-rank the articles by their authorities scores in descending sequence.

However, HITs algorithm has some limitations like advertising links effect, the hubs may provide links to more general pages, potential drift from main topic, etc. We can combine HITs algorithm and PageRank method or use some machine learning techniques to improve the performance.