**Q1**

A user submits a query q, to an information retrieval system which returns a ranked list to the user. Given the top k ranked documents.

1. Explain how, in terms of IR evaluation, one could measure the quality of this answer set. Discuss any limitations of this approach. (6 marks)

**How**:

Given the returned top K ranked documents, we can plot precision recall graphs.

* Because it is impractical to generate a precision-recall pair for every document in the ranked list (need to iterate through a very large list), we usually collect a limited set of precision-recall points. Typically, 9 points are collected corresponding to recall values of 10%, 20%, 30%...90%.

The scores of recall and precision are calculated as below: Let A be the returned answer set; let R be the relevant documents to query q.

Precision= Recall=

* We descend through the ranked list until the next recall point is reached, and then record the precision at this point to generate a precision-recall pair. (Note, this may involve interpolating (插值) between two recall points)
* We can use this metrics to capture the quality of the answer set. An ideal system, for a recall value of 1, the precision would also be 1, i.e. all relevant documents were ranked above all non-relevant documents. However, most of the time, users need to trade off between recall and precision according to domains and information need.

**However,** a precision-recall graph for one query against a system does not give us a meaningful insight into the true performance of the system, it is useful to plot the average of a number of queries runs against the system.

The average precision for the various recall values is calculated (given N queries):

If we compare several systems across many collections, the resulting large set of graphs can be unwieldy (笨拙的). Many methods can handle this like mean average precision.

* Summarise the list of precision-recall pairs by averaging the precision values.
* The mean of this across all queries is referred to as the MAP of the system. (SUM(pi)/n)

Besides, the use of the harmonic mean has been proposed as a means to combine both precision and recall into the one score:

Or E-measures:

**Limits:**

* In many domains we do not have available test collections and even in domains where we have large collections, meaningfully calculating recall is not possible as the collection is normally too large.
* These metrics are useful in capturing the usefulness of the returned set for a given query. However, they are useful for one off ad hoc queries (临时查询) but are inadequate in capturing the complexity of an interactive session.
* This method is based on the assumption that the set of relevant documents for a given query is the same for every user. But in reality, whether a document is “on-topic” or is relevant to a user’s information need should be distinguished.

1. Explain how the query q could be modified using evidence from the returned set to improve performance. Outlines any potential limitations of this approach. (10)

**How**

We can use local analysis to automatically expand query without explicit user feedback

1) In local analysis, a term-term correlation matrix Mi,j is created to quantify the connection between term i and term j. There are many different means to develop the correlations.

* Association Clusters

Calculate the number of co-occurrence of term i and term j in the returned document collection using:

* The above method only considers whether two terms co-occur in a document but does not consider their positions within a document. To improve this, we can use **metric clusters** using dist(ti, tj).

Where is the distance of these two terms in the same document.

* Moreover, the above methods do not consider the context of the query. **Scalar Clusters** can be applied here which pay attention to the neighbourhoods of terms. If two terms have similar neighbourhoods then there is a high correlation between them. In this approach, similarity can be based on comparing the two vectors representing the neighbourhoods using the cosine similarity measure. Use this to define a term-term correlation matrix and the procedure continues as before.

2) Then, we can develop an association cluster for each term ti. Given the ith 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 ti. 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.

3) Add these new terms to expand the original query. Also, may take all terms, or those with the highest summed correlation end itemize (结束项目).

**Limits:**

* Only re-retrieval the returned answer set, thus, may lack diversity of new terms added; Also, the improvement of performance cannot be guaranteed if the answer set is too small. Searching through the whole collection set may bring a higher recall;
* Term ambiguity may introduce irrelevant statistically correlated terms, leading to topic-drift problem.

1. Given the original query q and modified query q’, discuss in your words, how you might predict which query is likely to perform better. (9) **附加describe term query difficulty**

**Diagram

Description automatically generated**

Prediction of query performance prediction is also called query difficulty. The basis is how well the topic of the user’s query is covered by retrieved documents. Existing methods of predicting query performance can be divided into two categories: pre-retrieval methods and post-retrieval methods.

* Pre-retrieval pre-dictors utilize the static information of a query, which can be computed before retrieval. 1) **Linguistic approaches**: use NLP approaches to analyse query; use external sources of information to identify ambiguity. 2) **statistical approaches**:
  + Take into account the distribution of the query term frequencies in the collection (e.g., consider idf and icf of terms);
  + Take into account *specificity* of terms: queries containing non-specific terms are considered difficult;
  + Consider term relatedness: if query terms co-occur frequently in collection, we expect good performance.
  + Query scope: what percentage of documents contain at least one query term, if a lot then this is probably a difficult query.
  + Simplified query scope: measures difference between language model of collection with language model of query.
* Besides 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** that 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. Compare language model induced from answer set and one induced from the corpus/collection. This is related to the 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 same query to many systems over same collection; divergence in results tells us something about difficulty of query.

3) **score analysis** based methods that analyze the score distribution of results.

* difficulty can be measured based on distribution of values; cluster hypothesis is supported.
* can look at distribution of scores in answer set and document set and attempt to gauge difficulty.
* relatively simple measures shown to be effective.

**Describe query Difficulty**

Most IR systems exhibit large variance in performance in answering users’ queries. These can be caused by the query itself (ambiguous terms), vocabulary mismatch problem or missing content queries, as well as the robustness problem in IR. For example:

- failure to recognise all aspects in the query

- failure in pre-processing

- over-emphasis on a particular aspect/term

- query needs expansion

- need analysis to identify intended meaning of query (NLP)

- need better understanding of proximity relationship among terms.

**Q2**

1. Outline with an appropriate example, a suitable indexing strategy to deal with both Boolean and proximity queries. (8)
2. Outline a suitable data structure to allow searching for the presence of terms and prefixes of terms in passage of text. Illustrate, with a suitable example, how the data structure operates. (9)
3. Describe in your own words, with reference to any well-known term weighting scheme, the main constituents of a good weighting scheme. (8)

A good weighting scheme comprises local factors, global factors (collection wide) and query related features:

sim(q,d)=

Where TF(D) is the normalized term frequency in a document; gWt(C) is the global weight of a term across a collection; qWt(Q) is the query weight of a term in Q. For example, BM25/Okapi weighting scheme penalizes the long document, high term frequency and the words in a large number of documents, providing high retrieval accuracy.

Text

Description automatically generated



An axiomatic approach to IR (Fang and Zhai 2005) has been developed which refines a number of constraints (axioms) (Fang et al. 2004) to which all good weighting functions should adhere.

* Constraint 1: adding a new query term to the document must always increase the score of that document.
* Constraint 2 : adding a non-query term to a document must always decrease the score of that document.
* 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 a factor of thirty.
* Constraint 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.

**Q3**

1. Recommender systems are used to generate recommendations for users on unseen items. Collaborative filtering is one such approach. Explain the main stages of collaborative filtering in one such approach. Explain the main stages of collaborative filtering and illustrate this approach by generating a recommendation for item “Oh Mercy” for user “Jack”. (10)

* Build a user-item matrix based on the datasets of user, item, and ranking.
* Calculate the similarity between each user. (i.e. Pearson Correlation method)
* Form groups or neighborhoods of users who are similar. (i.e. best-n correlation)
* In each group, make recommendations based on what other users in the group have rated. (i.e. compute weighted average of user ratings using the correlations as the weights)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Dytan | Desire | Saved | Tempest | Oh Mercy |
| Jack | 1 | 5 | 3 | 3 |  |
| Lily | 1 | 4 |  |  | 4 |
| Rosemary | 4 | 4 | 3 |  | 5 |
| Rita | 1 | 4 |  |  |  |

* Calculate similarity of Jack and each other user using Pearson Correlation method.

Mean(Jack)=3

Mean(Lily)=3

Mean(Rosemary)=4

Mean(Rita)=2.5

We can modify the user vectors based on the distance between the ratings and the average rating:

Jack=(-2,2,0,0,0)

Lily=(-2,1,0,0,1)

Rosemary=(0,0,-1,0,1)

Rita=(-1.5,1.5,0,0,0)

Now, calculate similarity:

Sim(Jack, Lily)=(4+2)/sqrt(8\*5)= 0.9486833

Sim(Jack,Rosemary)=0

Sim(Jack,Rita)=(3+3)/sqrt(8\*9/2)=1

Thus, Rita is the most similar user to Jack, then is Lily and the last one is Rosemary.

* Given above, choose Rita and Lily as the similar users (neighborhoods) with Jack.

We can predict Jack’s rating to “Oh Mercy” based on Rita and Lily’s ratings. However, Rita does not rate “Oh Mercy”, so we can just see Lily.

* Compute the weighted average of ratings using the correlations as the weights. (similarity1\*rate1+similarity2\*rate2…)/SUM(similarity). Because the similar user here is only Lily, so the predicted value is:

0.9486833\*4=3.794733约等于4

方法二：

Text, letter

Description automatically generated

Rating of Jack=3+(4-3)\* 0.9486833/0.9486833=4

1. In many domains, in addition to collaborative ratings, extra information is often present and may prove useful in the provision of useful recommendations. For example, in the domain of music recommendations, we may have information on each album being rated. So, in addition to each rating present, we have for each album the following set of attributes: year, producers, musicians, genre, sub-genre, songs.

Outline an approach where this extra information can be used to hopefully improve the performance of the system. (8)

**Answer:**

Collaborative filtering has many issues like data sparsity problem and cold start problem. For example, it is unable to provide accurate recommendations when users and items have few ratings, resulting in reduced coverage. However, we can extract information from items and users and finding regularities in the content. Incorporating the content filtering and collaborative filtering can improve IR performance.

We can use user profile information such as gender, postcode, occupation, and their tastes, preferences to more accurately cluster similar users and identify their preferred items; Or we can use knowledge about how a particular object satisfies the user needs from the item description or user judgements to more accurately group similar items or match the target users. These need some NLP techniques because we need to extract text information. The information is extracted and represented as keywords with weights. And then, similarity between the user profile vector and the item feature vector can be calculated based on cosine angle or some traditional machine learning models. Hence, we can generate user-item matrix and user-user matrix based on the above content information.

As we know, collaborative filtering creates a user-user matrix based on history ratings of users and calculate user similarity based on the vectors of rating representations. To add content-based filtering, when generating a user-user or user-item similarity, we can combine the similarities generated by content filtering and collaborative filtering together with suitable weights respectively. For example, assign equal weights to content and collaborative filtering at the beginning. 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.

**另一种答案作为参考：**

We can combine content filtering and collaborative filtering.

* Step1: Firstly, use collaborative filtering to group similar users by their ratings. Creating a user-user matrix. If some songs are highly favorited by similar users, then these songs can be recommended to a user who also belongs to this group. We can set “k” as a threshold which denotes the top k songs can be recommended.
* Step2: Secondly, based on the attribute set of a song <year, producers, musicians, genre, sub-genre, songs>, we can cluster similar songs and generate a song-song matrix. Thus, we can know which songs have high similarity to the user’s favorites.

Besides, based on user’s profile or search history, the system can recommend the songs which have similar attribute description.

* Step 3: combine the results from collaborative filtering and the analysis of album attributes. For example: rating of item=α\*Collaborative filtering score + β\*Attributes Score. We can select the top N music based on the above formula which have high rating cores.

1. In your words describe how you would measure the performance of your system designed in part b. (7)

**Experimental Approach for Testing**:

1. a known collection of ratings by users over a range of songs is decomposed into two disjoint subsets: The first set (usually the larger) is used to generate recommendations for songs corresponding to those in the smaller set.
2. These recommendations are then compared to the actual ratings in the second subset.
3. 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.

(Accuracy is usually presented as the mean absolute error (MAE) between ratings and predictions.)

**Q4**

A company has a large set of scientific articles (each of which contains a title, abstract, authors, key words, year of release, main body of the paper and a bibliography.)

1. Suggest a means to measure the similarity between two documents based on (8)

* Content of the document
* Authors listed
* Bibliographies
* Content, authors and bibliographies

**Answer:**

The tf-idf weighting scheme can be used here. However, in this case it should be modified because the scientific articles have some attributes. For example, the terms which appear in the title, keywords, abstract, authors, bibliographies have higher resolving power. We should involve these factors to the tf-idf weighting scheme.

Let the weight of a term appearing in the content of a document be tf(c)\*idf; The weight of a term appearing in the authors listed be αtf(a)\*idf; The weight of a term appearing in the bibliographies be βtf(b)\*idf. Then the weighting scheme is function of the above:

Wij=f(tf(c)\*idf, αtf(a)\*idf, βtf(b)\*idf) (where β>α>1)

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

Wij= tf(c)\*idf+ αtf(a)\*idf+ βtf(b)\*idf

The vector of an article j can be denoted as:

Dj=(W1j, W2j, W3j, …. Wnj)

The similarity of two documents can be generated by calculating the cosine of the angle of their vectors.

1. The company wishes to rank all papers that are relevant to a given query and to then re-order the papers in the answer set according to how authoritative or influential the papers are. Outline an approach that could be used to give a suitable solution for this requirement. (10) **附加(limits)**
2. The weight of a term k in the query is Wkq = tf(k,q)\*idf, thus, a query can be denoted as:

Q=( W1q, W2q, W3q…, Wnq)

1. The similarity of a document and a query is sim(Dj, Q). The articles are then ranked descendingly by their scores and the top p articles are the answer set.
2. Use HITs algorithm to re-rank the returned articles.

* The answer set in the above step is the root R. We initialize base set S to R. And add to S all articles pointed to by any paper in R; Add to S all pages that point to any page in R.
* Assign to each article p∈ *S:*

an authority score: *ap* (vector***a***)

ahub score*: hp* (vector***h***)

Initialize all *ap = hp* to some constant value. (i.e.let this constant be 1)

Here, the authority score denotes the 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.

* Define M to be the adjacency matrix and make Mij =1 where .
* MTM can generate the authority vector ***a***. And MMT can return the hub vector ***h***.

Update the authority and hub scores:

***a←*** MTM ***a***

***h←*** MMT ***h***

* Iterate until convergence and the final authority and hub scores are generated.
* Re-rank the articles by the authorities scores in descending sequence.

**Limitations of HITs algorithm:**

**There exists problems with identifying authoritative pages:**

•authoritative pages do not necessarily refer to themselves as such

•many links are purely for navigational purposes

•advertising links

**Limitations of link only approach:**

1. on narrowly focussed query topics, there may not be many exact references and the hubs may provide links to more general pages
2. potential drift from main topic. All links are treated as being equally important. If there is a range of topics in a hub, the focus of the search may drift
3. timeliness of recommendation is hard to identify
4. sensitivity of malicious attack
5. edges with wrong semantics
6. Outline a suitable approach to cluster these documents in the collection into useful sub clusters that may be of use in user search tasks. Briefly list and limitations of the approach. (7) potential use of clustering

**Answer**:

**Potential use of clustering**:

Clustering involves the task of grouping data points into homogeneous classes or clusters. So that items in the same cluster are as similar as possible and items in different classes are as dissimilar as possible. All applications of clustering in IR are based (directly or indirectly) on this cluster hypothesis. Van Rijsbergen’s original wording: “closely associated documents tend to be relevant to the same requests”.

Clustering can be used to improve search recall: Cluster docs in collection are a priori; When a query matches a doc “d”, also return other docs in the cluster containing “d”.

**K-means algorithm**

Clustering is a kind of unsupervised learning where clusters are inferred without input.

Here, we can use K-means algorithm to cluster the documents.

* In K-means method, the documents are represented as vectors and the relatedness between vectors is calculated by Euclidean distance: (sqrt(x1^2+x2^2+…)).
* Each cluster is denoted by a centroid:

(Where is a cluster, α is the vector of a document)

* The objective is to minimize the average squared difference from the centroid.
* We try to find the minimum average squared difference by iterating two steps:

− Re-assignment: assign each vector to its closest centroid.

− Re-computation: re-computer each centroid as the average of the vectors that were assigned to it in re-assignment.

**The pseudocode is as follows**:

k-means({}, K) #x means vectors of all documents in the answer set, and K denotes K clusters.

) #select random seeds from the documents set.

**For** k ←1**to** k

**do** ←

**while** stopping criterion has not been met #stopping criterion could be reaching the lowest RSS

**do** **for** k ←1**to** k

**do** ←{ } #create a set to store all the documents assigned to this cluster

**for** n ←1**to** N

**do** j←argmin | | #get the smallest Euclidean distance j

← #reassignment of vectors

**for** k ←1**to** k

**do** ← # re-computation of centroids

**return** { }

(RSS=sum of all distances between documents vectors and the closest centroid)

**Limits:**

* Convergence does not mean that we converge to the optimal clustering. This is the great weakness of K-means. For example, if we start with a bad set of seeds, the resulting clusters can be poor.
* Random seed selection is not very robust: It’s easy to get a suboptimal clustering.
* The number of k is hard to determine.
* Using RSS as a stopping criterion will always choose K=N clusters which is not suitable.

Thus, we can add a penalty for each new added cluster and trade off cluster penalties against average squared distance from centroid