IR1 Sample Exam

1 Instructions

0.0 points · 1 question

(omitted)

2 Text preprocessing and indexing

6.0 points · 3 questions

Given a collection of text documents, you build two inverted indices:

- Index A -- no text preprocessing
- Index B -- stemming

Consider the total size (on disk) of **inverted lists** in each of these indices.
(Note, that the size of the vocabulary is **NOT** considered in this question).

In each of the following cases, how does the total size of inverted lists in index B compare to the total size of inverted lists in index A?

Please explain your answers in detail. Statements like "A < B" without explanation will result in 50% of the grade.

Description

a The indices contain document ids.

2.0 points · Open question

+2 points

Due to stemming, multiple words are merged into one (e.g., "run", "running" $\rightarrow$ "run"). So the inverted lists for such words are also merged. The size of the merged list is almost always smaller than the total size of all lists because the intersection between those lists is usually non-empty.

So $\mathbf{B} < \mathbf{A}$.

b The indices contain term frequencies.

2.0 points · Open question

+2 points

The same as above happens. But now the term frequencies for "run" and "running" are summed. Then it is a question of whether the sum of tfs occupies less space on disk/RAM than separate tfs or not.

It is likely that $\mathbf{B} < \mathbf{A}$, but this is not completely clear.

c The indices contain term positions.

2.0 points · Open question

+2 points

Since every term occurrence in a document occupies a separate space in the corresponding inverted list, merging two (or more) lists does not reduce their size (because all those separate occurrences remain separate and occupy the same amount of space).

So $\mathbf{B} = \mathbf{A}$.

3 Offline evaluation, metrics

6.0 points · 3 questions

For each of the following evaluation metrics, come up with one application/search scenario, where this metric is most suitable.

Explain, why it is most suitable for your application.

Description

a Precision@1

2.0 points · Open question

Grading description

Note for TAs: 1pt for the application and 1pt for the explanation.

+1 point

P@1 is used where only the first result is important and others are not.

+1 point

For example, homepage finding, navigation (to wikipedia, facebook, etc).

b Full/Total recall

2.0 points · Open question

+1 point

Full recall is used where all relevant document must be found.

+1 point

For example, patent search, medical search.

c ERR

2.0 points · Open question

+1 point

ERR is used in standard search scenarios, where a few relevant documents are important and they need to be ranked as high as possible.

+1 point

For example, search for information about something, search for movies/products, planning a trip, etc.

4 Offline evaluation, test collections

4.0 points · 3 questions

Consider a test collection, where relevance judgments are created using depth-k pooling with standard IR ranking methods, such as VSM, QLM, BM25, LSI, and LDA. Consider also that a completely new set of ranking methods was developed after the test collection had been created, e.g., neural ranking methods.

Description

a What problem may arise if you use the above test collection to perform offline evaluation of the new ranking methods?

2.0 points · Open question

+2 points

It is possible that the new ranking methods retrieve completely different documents compared to standard IR ranking methods (such as VSM, QLM, BM25, LSI, and LDA). If this is the case, the documents retrieved by the new ranking methods will not have relevance judgements, because they were neither retrieved nor judged during the depth-k pooling procedure. Since there are no relevance judgements available for the retrieved documents, these documents are assumed to be non-relevant. Offline evaluation using these unjudged documents will not reflect the real performance of new ranking methods.

b How can the test collection be modified to fix this problem?

1.0 point · Open question

+1 point

Run another round of depth-k pooling for the same collection using the new ranking methods, that is, collect top-k documents from the new ranking methods and judge them using human annotators.

c Is it possible to use the modified test collection to perform offline evaluation of standard IR ranking methods, such as BM25 and LSI? Explain your answer.

1.0 point · Open question

+1 point

Correct answer and explanation: The modified collection still contains original relevance judgements, so it can still be used to evaluate "old" ranking methods.

+0.5 points

Correct answer: Yes, but wrong/missing explanation

5 Term-based retrieval

3.0 points · 2 questions

You would like to use bi-gram language models for ranking.

Description

a Give an equation to compute the Query Likelihood Model $p_{bi}(q \mid d)$ in this case. Express $p_{bi}(q \mid d)$ in terms of probabilities. Take into account that a query may contain more than one term.

1.0 point · Open question

+1 point

Suppose

$$q = t_1 t_2 \dots t_n,$$

then

$$p(q \mid d) = p(t_1 \mid d) \cdot p(t_2 \mid t_1, d) \cdot \ldots \cdot p(t_n \mid t_{n-1}, d).$$

b For each probability used in the above equation, explain, how it is calculated for document d.

2.0 points · Open question

+1 point

$$p(t_1 \mid d) = \frac{tf(t_1, d)}{len(d)}$$

+1 point

$$p(t_2 \mid t_1, d) = \frac{tf(\{t_1, t_2\}, d)}{tf(t_1, d)}, \text{ etc.},$$

where $\{t_1, t_2\}$ is a sequence of terms t_1 and t_2

6 Semantic retrieval

2.0 points · 1 question

You use two low-rank approximations for LSI:

- LSI_1: $k = 1000$
- LSI_2: $k = 100$

Description

Explain, why total recall is higher for LSI_2 compared to LSI_1?

2.0 points · Open question

+1 point

For lower values of the parameter k, LSI representations of document become more general/less specific, because we remove more specific information and make vector representations shorter.

+1 point

More general representations allow to retrieve more documents, given a user's query. This, in turn, results in higher total recall.

7 Offline LTR

7.0 points · 5 questions

Consider the pairwise approach to offline Learning to Rank (LTR).

Description

a In pairwise LTR, preferences between two documents are modeled.
Consider the following model that predicts preferences for each pair of documents: $f(d_i, d_j) = P(d_i \succ d_j)$.

What is a practical problem with this model?

2.0 points · Open question

+1 point

Function f works on pairs of documents. This means that at a query time it has to be applied to all pairs of documents to predict preferences between them and to rank the documents.

+1 point

This process has complexity $O(N^2)$, where N is the number of documents. For large N this is too slow to rank documents given a query.

b How can this problem be solved?

1.0 point · Open question

+1 point

Use pointwise model $f(d_i) = s_i$ and pairwise loss $\mathcal{L} = \sum_{d_i \succ d_j} \phi(s_i - s_j)$.

c Pairwise LTR (e.g., RankNet) has a smooth differentiable loss function and, thus, can be optimized using gradient descent. Still, there is a fundamental problem with the pairwise LTR approach. What is this fundamental problem?

1.0 point · Open question

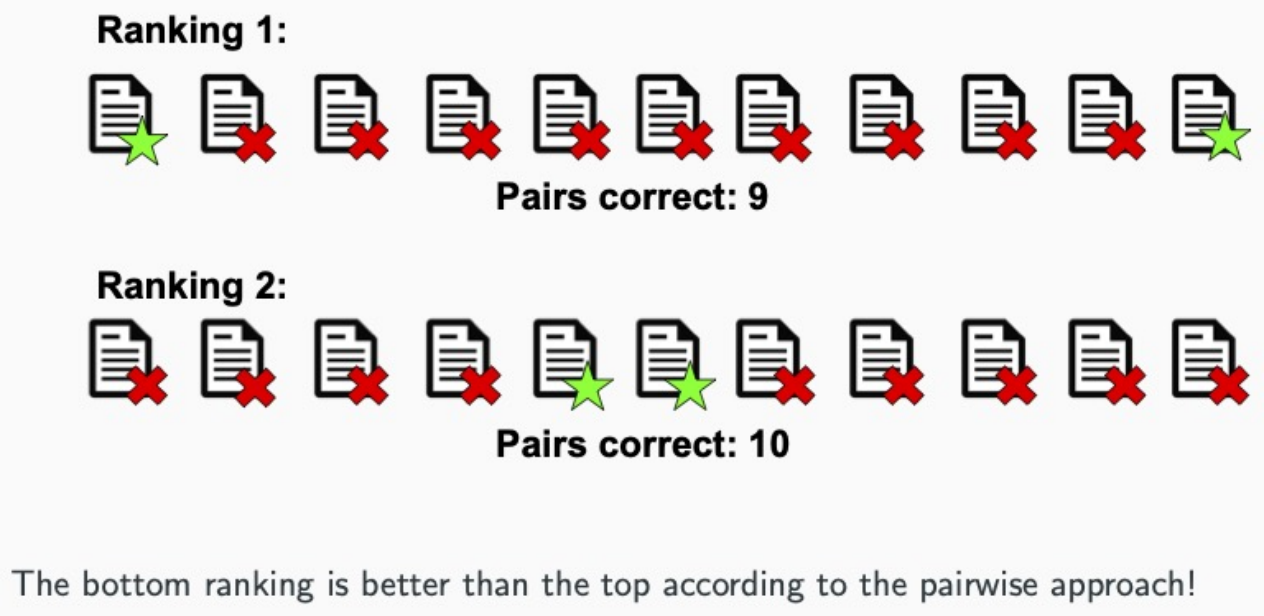
+1 point

Pairwise LTR optimizes the number of correctly ranked pairs of documents but does not optimize the entire ranking. Optimal placement of pairs may easily result in a bad ranking.

d Give an example ranking that illustrates the above fundamental problem. Explain your example.

2.0 points · Open question

+1 point



+1 point

The first ranking is preferred by users (because the relevant document is on the first position). The second ranking is preferred by pairwise LTR (because it has more correct pairs and fewer incorrect pairs).

e How does LambdaRank solve this problem?

1.0 point · Open question

+1 point

LambdaRank replaces the cost function inside the gradient with $|\Delta NDCG|$, where $\Delta NDCG$ is computed by swapping documents d_i and d_j in the ranking.

8 IR-user interaction

6.0 points · 3 questions

Consider using clicks for IR evaluation instead of relevance judgements.

Description

a Give two pros of using clicks instead of relevance judgements.

2.0 points · Open question

Grading description

Note for TAs: 1pt for every pro and con.

+1 point

Clicks are easy to collect (given a running search system).

+1 point

Clicks are many.

+1 point

Obtaining clicks is cheap (again, given a running search system).

+1 point

There is a good alignment between train and test user preferences/clicks.

b Give two cons of using clicks instead of relevance judgements.

2.0 points · Open question

+1 point

Clicks are biased (e.g., position bias).

+1 point

There may be no clicks.

+1 point

Clicks may happen on non-relevant documents.

+1 point

Clicks are ambiguous (i.e., their meaning is not always clear).

c What information about user search behavior can be added to clicks to improve their quality/reliability? How should clicks be combined with this additional information?

2.0 points · Open question

+1 point

We need to measure dwell time (time spent on a clicked result).

+1 point

Clicks with long dwell time better reflect the relevance of a result than those with short dwell time.

9 Counterfactual LTR

3.0 points · 2 questions

Consider counterfactual learning to rank (LTR).

Description

a Counterfactual LTR assumes a relation between a click on a document and the document's relevance. What is this assumption?

1.0 point · Open question

+1 point

A document is more likely to be clicked if it is more relevant.

b Give two situations where this assumption does not hold.

2.0 points · Open question

Grading description

Note for TAs: 1pt for every situation.

+1 point

Over/under estimation of relevance by the user, e.g., click-bait websites have titles that make people overestimate the relevance of the site, thus there may be many clicks despite little relevance.

+1 point

No need to click the document, e.g. just the title or the snippet of a website can answer a user's need, thus no longer requiring them to click on the link to be helpful.

+1 point

Dependencies on other documents in the ranking: if a previous result answers the user's need, there may be no need to continue looking at other documents.

+1 point

Misclicks or similar 'mistakes' by the user.

+1 point

Positional bias where the document is low in the rankings or 'trust' in the search engine where a user clicks the first document in the ranking.

10 Online evaluation

2.0 points · 1 question

Explain, why A/B testing requires more user interactions than interleaving (you can also explain the other way around, i.e., why interleaving requires fewer user interactions).

2.0 points · Open question

+2 points

A/B testing uses two sets of different users. So it needs to marginalize out potential behavioral differences between two populations. For that, it needs more data. Interleaving, instead, uses the same users for all experiments, so there is nothing to marginalize out.

11 Recommender systems

3.0 points · 2 questions

Consider the neighborhood-based approach to collaborative filtering. In this approach, a missing rating r_{ui} by user u for item i is calculated based on the k-nearest neighbors of user u. This is also know as a user-based approach, because we consider neighbors of the user u.

Description

a Give an equation for calculating rating r_{ui} for item-based approach, where we consider k-nearest neighbors of item i.

1.0 point · Open question

+1 point

Consider k-nearest neighbors of item i rated by user u: $M_u(i)$.
Then

$$r_{ui} = \frac{1}{|M_u(i)|} \sum_{j \in M_u(i)} r_{uj}.$$

b Given an item, explain, how to find its k-nearest neighbors.

2.0 points · Open question

+1 point

Each item is represented as a vector of ratings (some cells may be empty if no rating was given).

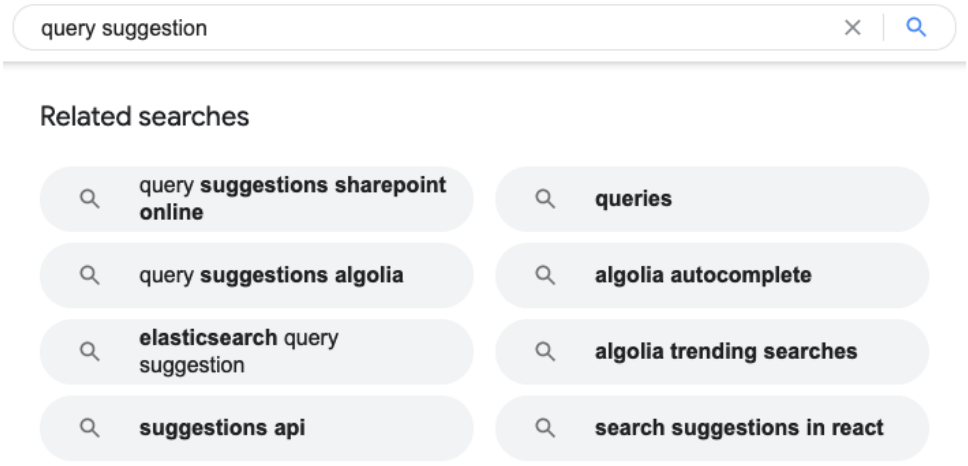
+1 point

The similarity between items can be computed in many different ways, e.g., using cosine similarity. We need to choose k most similar items to the given one.

12 Query suggestion

5.0 points · 3 questions

Query suggestion is a task, where, given a query q and a set of all possible queries $\mathcal{Q}$, one needs to rank $q_i \in \mathcal{Q}$ according to how likely a user is to submit q_i given that she submitted q. An example of *query suggestion* can be seen in the following figure:



Propose algorithms for *query suggestion*. The proposed algorithms should clearly explain how $q_i \in \mathcal{Q}$ are ranked with respect to q. For example, how the score for each q_i is calculated.

Note 1: Vague terms, such as "similar'', "close'', "higher'', etc., will only be given 50% of the points. Instead, please use specific terms, such as "<name_of_similarity_function>'', "distance of 2'', "larger by 1'', etc.

Note 2: The proposed algorithms should be implementable without any additional information. You can refer to any notion presented during the course without explaining it in detail, e.g., "inverted index'', "MAP'', "LSI'', "LTR'', etc.

Description

a Propose a content-based *query suggestion* algorithm that only uses the content of q and $q_i \in \mathcal{Q}$.

1.0 point · Open question

Grading description

Note for TAs: The proposed algorithms should be implementable without any additional information. If the algorithms are generally correct, but some details are missing, give 50% of the grade.

+1 point

Any content-based ranking method can be used here to calculate the similarity between the original query and q and query suggestions q_i.

b Suppose you have a query log which contains submitted queries (all those queries are in $\mathcal{Q}$), the order in which the queries are submitted, and the timestamp for each submitted query.

Propose an interaction-based *query suggestion* algorithm that only uses this query log (and does **NOT** use any content information).

2.0 points · Open question

+1 point

50% of points if not enough details. E.g., use query co-occurrence for query suggestion.

+2 points

Any answer of the following type is correct: count how many times q and q_i appear one after the other (in any order or even with a distance of N, where N is specified by the student); count how many times q and q_i are submitted within M seconds/minutes (where M is specified by the student). Rank q_is based on that.

c Suppose the query log also contains the returned search results for each submitted query and the corresponding click information.

Propose an interaction-based *query suggestion* algorithm that uses this click information (and still does **NOT** use any content information).

2.0 points · Open question

+1 point

50% of points if not enough details. E.g., use co-clicked documents for query suggestion.

+2 points

Count the number of co-clicked documents for q and q_i. Rank q_is based on that.