Answer all of the following questions. Each answer should be thorough, complete, and relevant. Points will be deducted for irrelevant details. Use the back of the pages if you need more room for your answer.

The exam should take you about 70 minutes to complete. The points are a clue about how much time we think each question should take to answer. We assume about 1.5 points per minute, so a 10-minute question is worth 15 points. Plan your time accordingly.

Good luck.
1. The following metrics have been commonly used in evaluating a ranked list:
   - P@N, the precision at threshold N in a ranked list
   - Average Precision (AP), the average precision at all the positions where relevant documents occur in a ranked list
   - Mean Reciprocal Rank (MRR), the inverse of the rank of first relevant document
   - Normalized Discounted Cumulated Gain (NDCG)

For each of the above metrics, discuss a situation when and why the metric is not a good choice for evaluating search engines, and which other metric(s) would be a better alternative. **Hint:** Discuss with respect to the nature of the query (e.g., the number of relevant documents for the query) or the preference of the user. **[16 points]**

**Answer**

- **P@N** is not a good choice when the query has only one correct answer because it is insensitive to the rank of the relevant document when retrieved, and not discriminative when not retrieved. **MRR** is a better alternative.

- **MRR** is not a good choice when the query has more than one relevant document, and the user want to see most of them, but the metric only focus on the first relevant document instead. **AP** and **NDCG** are better alternatives.

- **AP** (as well as **P@N** and **MRR**) is not a good choice when multi-grade relevance judgments are available for the query because it is only defined on binary judgments. **NDCG** or **NDCU** is a better alternative.

- **NDCG** is not a good choice when the reading cost and the redundant information in the ranked list is a concern. In particular, when the query has only one correct answer, **MRR** is a better choice than **NDCG**. In general, **NDCU** or Expected Utility is a better choice when the user is interested in more than one relevant documents.
2. What are the three most common types of information need on the web? Give an example of each type. What are the expected characteristics of search results for each type of information need? What would be the main types of information used for each type of information needs? [15 points]

**Answer:**

**Informational queries**
- Example: “new breast cancer treatments”, “INI admissions requirements”
- Search results should be topically related to the query. A person may want many relevant documents, and doesn’t care how many sites they come from. Diversity might be important in the search results.
- These queries are more likely to match text in the body of the document than in the title or inlink text, so sequential dependency model queries would be a good choice.

**Transactional queries**
- Example: “buy airline tickets”
- Search results should be a set of sites where the specified transaction can be accomplished. A person probably wants only a few pages, and each page should come from a different site. Diversity in the search results is probably not important.
- Inlink and title text is probably very important for these queries, because it is more likely to be similar to the queries that people type. PageRank might be especially important, for example, to favor popular sites for buying airline tickets.

**Destination queries**
- Example: “united airlines”, “jamie callan”
- The user only wants one search result, which is the named website. Diversity in the search results is probably not important.
- Inlink text, title text, and url text are probably all important. PageRank might be less important for these queries, because presumably there is just one right answer, and it might not be a popular site (e.g., Jamie’s web page). url depth is probably important (shallow urls are more likely to be home pages).
3. Suppose you have just been hired by a company that uses Lucene as its enterprise search engine. Your boss wants to improve search quality, but doesn’t want to replace Lucene. Describe what actions you could take to identify search quality problems, improve search quality, and measure the improvement. [12 points]

**Answer:**
There are many acceptable answers to this question. A few examples are shown below.

To identify search quality problems, you could examine search logs to find common queries (“head” queries) and uncommon queries (“tail” queries). You could look at clickthrough patterns for head queries to get information about whether the documents that people want tend to be ranked high in the ranked list, or lower in the ranked list. For all queries, or just tail queries, you could get people to judge search results (obtain relevance judgments). This information would enable you to measure Precision, MAP, and other popular metrics. Note that unless you search very deeply and/or pool results from multiple methods, you probably won’t have any idea about Recall.

You could also look at search log sessions and query reformulation patterns to get a sense of whether people are finding what they want on the first query, or having to issue multiple queries to find what they want. Looking at reformulation patterns might be also enable you to identify the “problem” queries.

To improve search quality, you might capture bag-of-words queries and reformulate them automatically prior to submitting them to the search engine. For example, you might transform them into sequential dependency model queries, or queries that use multiple text representations, to improve search results.

You might also work on improving the text representation of documents stored in the search engine. For example, you might tune the stopword list, or use field-based indexing, or other methods that would improve the matching of queries to documents.

You could also provide query suggestions, but this will only improve search quality if you can distinguish between past queries that were effective and past queries that were not effective.

To measure the improvements, you will need relevance judgments for a reasonably large set of sample queries. Clickthrough information is useful for popular queries, but not sufficient by itself. You can use relevance judgments to measure the system’s accuracy using metrics like Precision and MAP. You can also use measures of user satisfaction, e.g., average rank of first click, average session length (number of reformulations), abandoned searches, and heuristics that identify successful searches.
4. Suppose that you are using Indri to run a specialized search service. You would like to begin personalizing search results. During indexing, you use a text classifier to assign each document to 1 category. (There are 50 categories.) Each person’s preferences are learned by counting how often they click on pages in each category during a 3-month period. Thus, for each repeat customer, you know the \( p(\text{click} | \text{category}_i) \) across 50 categories. How would you use this information (with the Indri search engine) to produce rankings that are accurate for the query, and personalized? Justify your solution, and discuss its effect on queries of different lengths. [12 points]

Answer:
Use Indri’s default algorithm to retrieve the top 200 results. Then re-rank each document using \( p(\text{click} | \text{category}_i) \) as a prior probability in the ranking algorithm. In other words, 
\[
p(\text{click} | \text{category}_i) \times p(q | d)
\]
This fits well in the Indri retrieval model. It also has the advantage that personalization has a stronger effect on short (probably ambiguous) queries, and less of an effect on long (probably more specific) queries.

You could also use a weighted combination of the Indri score and \( p(\text{click} | \text{category}_i) \) for each document’s category. In other words, 
\[
(1 - \lambda) \times p(q | d) + \lambda \times p(\text{click} | \text{category}_i)
\]
This would probably work, but is weaker theoretically, and probably gives personalization too much effect unless \( \lambda \) is very small. Personalization has the same effect on short and long queries, which may not be desirable.
5. Suppose you want to support sequential dependency model queries in your web search service, because you know that they improve retrieval accuracy.

a) What query operators does the sequential dependency model require? [6 points]

**Answer:**
NEAR/n, UW/n, an operator for weighting different query elements, and an operator for unstructured bag-of-words queries. The latter could be AND, as it is for Indri, but it could also be SUM, for example if used with Okapi or the vector space.

b) Which of these retrieval models can support sequential dependency model queries: Vector space, BM25, and Indri. Justify your answer. [6 points]

**Answer:**
All three retrieval models support NEAR/n, UW/n, a weight operator, and unstructured bag of words queries, thus all of them can support sequential dependency model queries. Typically sequential dependency model queries are used with Indri, but there is no reason that they can’t be used with other retrieval models, too.
6. Three different uses of document structure were discussed in class: Fields, multiple representations of text content, and hierarchical structure.

a) Give an example of each type of document structure. [6 points]

Answer:
- Fields: Author, date, publication, patient name. The distinguishing characteristic is that the contents are relatively independent of each other, i.e., knowing the author name doesn’t provide any information about the date.
- Multiple representations of text content: For web documents, title, body, url, inlink. The distinguishing characteristic is that each representation describes more-or-less the same information, although perhaps with different levels of detail or reliability, and that each representation is relatively flat (not hierarchical).
- Hierarchical structure: XML documents, scientific papers. The distinguishing characteristic is that information is organized hierarchically, and the information in a parent element is likely to be topically related to the information in a child element.

b) Explain special treatment that a search engine might provide for each type of structure. Focus on treatments that are different for each type of structure. [9 points]

Answer:
- Fields: The contents of each field are considered independent sources of evidence. In a probabilistic model, most likely the query combines the evidence from each field using an AND operator.
- Multiple representations of content: The contents of each representation are considered different ways of describing the same information, i.e., the contents of each representation are not independent. In a probabilistic model, most likely the query combines the evidence from each representation using a weighted average (WSUM) query operator.
- Hierarchical structure: The contents of ancestor (e.g., parent) and descendent (e.g., child) fields are considered to cover related content. In a language modeling system, the maximum likelihood probability computed for a child may be smoothed using the maximum likelihood probability of one or more ancestor elements. The system may also allow the score of a child element (e.g., section) to affect the score of a parent element (e.g., document). An example used in class was retrieval of documents about breast cancer that include sections discussing treatments, e.g., #combine(breast cancer #combine[.section](treatment)).
7. Suppose that you work for a search engine that provides information about concerts, movies, festivals and related events. The index treats each event as a document; each document has several fields, such as event type, event title, location name (venue), and a 'body' field that includes the textual content from the whole document. The service has been running for 6 months and it has become very popular. Of course, since you planned ahead, you have been collecting logs for the search queries, the results that were displayed and also the ones that were clicked.

Your task is to build a component that rewrites queries and identifies the query terms related to the 'location' field. For instance, queries might be rewritten as follows:

"batman cinemark" \rightarrow #DEF( batman #LOCATION( cinemark ))
"Madonna consol energy center" \rightarrow #DEF( #LOCATION( consol energy center ) madonna )

where #DEF is the default operator, and #LOCATION searches for the query terms in the 'location' field.

a) How would you build this component? [10 points]

Answer:
A really simple bag of words approach would be the following:

Use the values of the location field to create a big document with all of them. For instance, the document will contain all the movie theaters and concert halls.

When the query is evaluated, every term or pair of adjacent terms can be matched against the big document and scored using tf-idf. The term or pair with highest score (above a certain threshold) is selected as argument for the operator.

b) Would this change improve precision or recall? Explain your answer. [8 points]

Answer:
It would improve precision. Instead of matching the 'body' field, the query terms would need to match only the contents of that field, so the results should be more specific.

c) How would you evaluate your component? What issues do you think that you will find? How would you address them?

Answer:
AB testing. The experiment consists of showing the results returned with and without the component and monitor the click-through rate.

Advantages:
• It could take less time and the experiment could be repeated many times, without requiring too much effort for additional tries.
• It is possible that the results are personalized or depend on the context of the search (e.g time or location of the user), so it is hard to create a large test collection with relevance judgements.

Possible Issues:
• Find the right types of queries to run the experiment (eg. run them for queries that have more than 10 results, run them on queries that yield different results under the two components).

• Find a representative group of users (location, mobile or PC, time of the query)