Student Name: _____

Andrew ID: _____

Seat Number: _____

Midterm Exam
Search Engines (11-442 / 11-642)
October 20, 2015

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.

Calculators, phones, and other computational devices are not permitted. If a calculation is required, write fractions, and show your work so that it is clear that you know how to do the calculation.

Advice about exam answers...

- Sometimes an answer says "I would use <technique> to do <x>". That answer shows that you remember a name, but it does not show that you remember how the technique works, or why it is the right tool for this problem. Give a brief description of how the technique works and why it is the right tool for this job. If the technique needs other information, explain where the information comes from.

.

1 Evaluation

Suppose that a large health provider has a website with a search engine that allows patients to find information about staying fit, eating well, diseases, treatments, and tests. The search engine receives about 35 queries per week. The company doesn't have data for evaluating the accuracy of the search engine, and doesn't know its accuracy. Describe how you would evaluate the accuracy of the search engine. Be clear about the method you would use, the data it would require, how much data it would require, and how you would get the data. Explain why your method is the right choice for this problem. **[15 points]**

Answer

The search engine doesn't receive much traffic, so there isn't enough click data and there isn't enough traffic to use interleaved testing. The Cranfield methodology is the best choice in this situation. Start by randomly sampling 50-100 queries from the query log. Develop written information needs to describe what each query is about. If possible, index the documents with several open-source search engines, run each query against each search engine, and pool the results for each query to form a pool of documents to be assessed. If it is not possible to use several open-source search engines, then create several (e.g., 5) variants of each query, run each variant against the search engine, and pool the results to form a pool of documents to be assessed. The size of the pools is determined by the available budget, and the nature of the problem; probably top 100 is sufficient for this task because most web-site visitors won't search very deeply into the results. Sort the pool of documents for each query into a random order. Have someone assess the results, either on a binary scale (relevant vs. non-relevant) or on a 5-point scale. Use several metrics (P@n, MAP, NDCG) to assess the search engine quality.

2 Document Structure

Two queries and two structured documents are shown below. For each query, explain i) what the query ranks; ii) whether information from different paragraphs is combined, and if so, how; iii) how scores are calculated (do not write formulas), and iv) whether it gives a higher score to $Document_1$ or $Document_2$ and why (assume that they have the same overall length). **[16 points]**

$Query_1$: #and (silicon.paragraph valley.paragraph bubble.paragraph)
$Query_2$: #and[paragraph] (silicon valley bubble)

$Document_1$		$Document_2$
My Favorite Bubbles and Valleys	title	Is it Finally Over?
…. An Irreverent Journey of Faith happens on Bubble girl ….	$paragraph_1$	There are 4 Signs That Silicon Valley's Tech Bubble Is Bursting…..
…. My leakiness is all the more evident against the shriveled, brown autumn landscape of Valley. ……. There are many minerals around their hometown, silicon, iron… … silicon gulch …	$paragraph_2$	Tourists are taking the bus out of Silicon Valley and One of the big reasons for the blowing up of the latest bubble has been big money "tourists"…..
… bubble, bubble, toil and trouble … … crossing the valley of death …	$paragraph_3$	…. proliferation of unicorns in Silicon Valley is another sign of Bubble ….

Answer:

$Query_1$ ranks documents. $Query_2$ ranks paragraphs.

$Query_1$ combines the information from all paragraph elements into a single inverted list. Then it uses the default document-scoring algorithm (e.g., MLE with two-stage smoothing) to calculate a score for the document.

$Query_2$ does not combine information from different paragraphs. Instead, a score is calculated for each individual paragraph element using a mixture model scoring algorithm that is a weighted average of the MLE score for the element, document-level smoothing, and corpus-level smoothing.

$Query_1$ ranks $Document_2$ ahead of $Document_1$ because $Document_2$ has more occurrences of 'silicon'; the two documents have the same number of other query terms. $Query_2$ doesn't rank documents.

3 Retrieval Models

Inverse document frequency (idf) and the Robertston Sparck-Jones (RSJ) weight are used to accomplish a similar purpose. Write both formulas (and define your variables). Explain their purpose (i.e., why their behavior is desired or important). Describe how their behavior differs. For example, if you replaced the BM25 RSJ weight with idf, what would be the effect on the document ranking algorithm? **[15 points]**

Answer:

idf (t): Any of the following formulas are acceptable:

$$idf(t) = \log\left(\frac{N}{df_t}\right) \qquad idf(t) = \log\left(\frac{N+1}{df_t}\right) \qquad idf(t) = \log\left(\frac{N}{df_t}\right) + 1$$

RSJ(t): Both of the following formulas are acceptable:

$$RSJ(t) = \log\frac{\mathrm{N} - \mathrm{df_t} + 0.5}{\mathrm{df_t} + 0.5} \qquad RSJ(t) = MAX\left(0, \log\frac{\mathrm{N} - \mathrm{df_t} + 0.5}{\mathrm{df_t} + 0.5}\right)$$

where N is the number of documents in the corpus and df_t is the number of documents that contain t.

Both formulas favor words that are rare in the corpus. Rare words are better able to discriminate relevant documents from non-relevant documents.

The RSJ weight has a stronger reward for rare terms and a stronger penalty for frequent terms. Words that occur in more than half of the corpus get a weight of 0. If the RSJ weight is replaced with idf, the penalty for frequent words is reduced – especially for words that occur in more than half of the documents.

4 Document Representation

Search engines use shallow language analysis and heuristics to convert a document a bag-of-words (index terms). Why is lexical processing critical to the performance of search engines? Describe four lexical processing methods. For each method, give examples to show its advantages and disadvantages, or the tradeoff of using this method. **[16 points]**

Answer

Lexical processing transforms a document from natural language to a representation of meaning that a search engine can process quickly when a query is received. This transformation must produce a representation that is simpler, but that also retains all of the important meaning of the document. Often poor accuracy is due to a poor text representation.

Types of lexical processing:

- Case conversion: Apple -> apple. Advantages: Improves recall, matches more queries. Disadvantages: Apple may be used as a company name, while apple will be considered as a kind of fruit.
- Stemming: apples -> apple. Advantages: Improves recall, matches lexical variants that have the same meaning. Disadvantage: May conflate words that a person would consider different (e.g., 'execute' and 'executive') or different in a particular situation (e.g., the company 'apple' and 'apples').
- Stopword Removal: the, of, a. Advantages: Discard meaningless word, reduce index size, improve accuracy. Disadvantage: "to be or not to be" after removal, it makes some queries difficult to satisfy. If there is no disk limit, we could store stop words in our index, and we could specify rules that if stopwords are more than half the terms, we still keep it when matching queries.
- De-compounding: computer-virus - > computer, virus: Advantages: Provides a more accurate representation of the document, enables a broader range of queries to match. Disadvantage: n-grams like "roe v. wade" will be meaningless.
- Other possible answer will be: phrase, POS

5 Relevance Feedback

What is relevance feedback? Provide two reasons that it is it not used in web search engines. Describe a task or environment where relevance feedback could be expected to be used effectively. **[14 points]**

Answer

Relevance feedback is a supervised machine learning approach to creating improved queries. It learns a larger bag-of-words and term weights from example documents. It is not used much in web search engines because i) users dislike giving feedback, and ii) there is a high risk of learning a poor query when a user only judges a small number of relevant documents. Relevance feedback could be expected to be effective in high-Recall tasks such as legal or patent search where users are willing to judge many documents to produce a query that has a high likelihood of finding many relevant documents.

6 Heaps Law

Write the formula for Heaps Law (define your terms). Give one practical example of its use (i.e., a situation where it would be useful). **[10 points]**

Answer

$$V = KN^{\beta}$$

where

- K is a constant. Usually $10 \leq K \leq 100$.
- N is the number of word occurrences in the corpus
- β is a constant. Usually $0.4 \leq \beta \leq 0.6$ for English.

Heaps Law is useful for predicting the size of a term dictionary before building a system.

7 Indexing

Describe Top-docs (Champion) Lists. What is their purpose, how are they constructed, and how many terms have a Top-docs list (give an intuitive explanation, not a formula)? What is an advantage and a disadvantage of using top-docs lists? What kinds of query operators are a good match to Top-docs lists, and what kinds are a poor match? **[14 points]**

Answer:

A Top-docs list is a type of inverted list that contains only the best documents that match a query term. Top-docs lists are constructed by sorting the full inverted list on some metric, for example tf, p_smoothed (t|d), or PageRank (d), and then selecting just the top N documents. Typically a Top-docs list is designed to fit into one or two operating system pages. Given Zipfs Law, typically only a small percentage of terms is frequent enough to have a Top-docs list. The advantage of Top-docs lists is that they are much faster to process than full inverted lists. The disadvantage is that they don't contain all documents, so the document ranking may be less accurate than a ranking that would have been gotten with full lists. Score operators except RankedBoolean #AND, because they can match a document even if some arguments do not match. Bad match: Boolean #AND operators because they only match if all arguments match. #NEAR and #WINDOW are off topic because Top-docs lists don't contain positions.