

**Student Name:** \_\_\_\_\_

**Midterm Exam**  
Information Retrieval and Web Mining (15-493)  
October 14, 2008

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.

**Short Answer** (5 minutes or less per question):

1. What is the idf of a term that occurs in every document? Justify your answer. Compare this weight to the use of stopword lists. [7 points]

**Answer:**

The idf of a term that appears in every document is 0. idf is defined as  $\log(N/df)$ . When  $df=N$ ,  $idf = \log(1) = 0$ . In some retrieval algorithms, for example the vector space lnc.ltc algorithm, an idf of 0 would cause a term to contribute 0 to a document's score, thus the term would be indistinguishable from a stopword (i.e., a term that was discarded from the index).

Some people defined  $idf = \log(N/df)+1$ , which is a form that we discussed in class. This form is preferred when the goal is to prevent idf from ever being 0. In this case, a term that appears in every document contributes only a small weight to a document's score, but it does contribute something, whereas a stopword contributes nothing.

2. Normalized Discounted Cumulative Gain (NDCG) is sometimes the preferred metric for evaluating web search engine performance because it focuses attention on the top of the rankings (e.g., first page of results), and allows multi-valued relevance assessments. Calculate NDCG for the ranking below, where  $R_i$  indicates a relevant document with value  $i$ . (Show the calculation; you do not need to produce a single number.) Assume that non-relevant documents have value 0, and ignore the normalizing constant. [8 points]

1	2	3	4	5	6	7	8	9	10
<input type="text" value="R&lt;sub&gt;2&lt;/sub&gt;"/>	<input type="text"/>	<input type="text" value="R&lt;sub&gt;1&lt;/sub&gt;"/>	<input type="text" value="R&lt;sub&gt;2&lt;/sub&gt;"/>	<input type="text" value="R&lt;sub&gt;1&lt;/sub&gt;"/>	<input type="text" value="R&lt;sub&gt;1&lt;/sub&gt;"/>	<input type="text"/>	<input type="text"/>	<input type="text" value="R&lt;sub&gt;1&lt;/sub&gt;"/>	<input type="text"/>

**Answer:**

$$NDCG = Z_k \sum (2^{R(i)} - 1) / \log(1+i)$$

If the normalizing constant  $Z_k$  is ignored, the computation is equivalent to DCG (i.e., no normalization).

$$\begin{aligned} DCG &= ((2^2-1) / \log(1+1)) + ((2^1-1) / \log(1+3)) + ((2^2-1) / \log(1+4)) + \\ &\quad ((2^1-1) / \log(1+5)) + ((2^1-1) / \log(1+6)) + ((2^1-1) / \log(1+9)) \\ &= (3 / \log 2) + (1 / \log 4) + (3 / \log 5) + \\ &\quad (1 / \log 6) + (1 / \log 7) + (1 / \log 10) \end{aligned}$$

3. Give a one-sentence description of the three most common types of information needs encountered in web search. [9 points]

**Answer:**

Informational: Find information on a topic, e.g., information about breast cancer.

Transactional: Find a site to carry out a transaction, e.g., to buy DVDs.

Navigational: Find a specific web location, e.g., a person's home page. I gave partial credit for "home page" search, although that is just one example of navigational search.

Full credit required either the names of the types above (informational, transactional, etc), or a description that was very clearly one of the above types. Vague descriptions that more-or-less match one of the above types didn't count.

4. Write the formula for the vector space ranking algorithm known as lnc.ltc. Define your variables. [6 points]

**Answer:**

l:  $\log(tf_i + 1)$  or  $\log(tf_i) + 1$ . Both were acceptable.

n: No weight /normalization (i.e., 1.0)

c: cosine normalization

t:  $\log(N / df_i)$

$tf_i$ : How often the  $i^{\text{th}}$  query term occurs in the document.

$qtf_i$ : How often the  $i^{\text{th}}$  query term occurs in the query.

$df_i$ : The number of documents that contain the  $i^{\text{th}}$  query term.

N: The number of documents in the collection.

$$\frac{\sum d_i \cdot q_i}{\sqrt{\sum d_i^2} \cdot \sqrt{q_i^2}} = \frac{\sum (\log(tf_i + 1)) \times \left( \log(qtf_i + 1) \times \log \frac{N}{df_i} \right)}{\sqrt{\sum (\log(tf_i + 1))^2} \times \sqrt{\left( \log(qtf_i + 1) \times \log \frac{N}{df_i} \right)^2}}$$

5. Inverted lists can be sorted by document id (as in your homeworks) or by term frequency (as in champion lists). Both approaches are popular. Compare the two approaches, paying particular attention to i) the computational advantages and costs of each method, and ii) the effect on typical inverted list compression algorithms. [15 points]

**Answer:**

When inverted lists are sorted by document id:

- Works well for all query operators.
- Assumes that the system will process the entire inverted list, which can be costly (in I/O, in computation) if query terms are very frequent.
- Document scores are guaranteed to be accurate.
- Combining information from different lists to produce a document score can be done very efficiently, because each inverted list is in the same order.
- The delta encoding (“store the gaps”) compression algorithm can be applied to document ids, because they increase monotonically.

When inverted lists are sorted by term frequency:

- Works only for document score query operators (e.g., AND, OR, AVERAGE, SUM).
- Assumes that the system will only process partial inverted lists, which can be very efficient (in I/O, in computation) if query terms are very frequent.
- Document scores may be less accurate, although they are usually a good approximation.
- Combining information from different lists to produce a document score requires a more costly and complex algorithm because inverted lists are not in the same order.
- The delta encoding (“store the gaps”) compression algorithm can be applied to term frequencies, because they decrease monotonically. Compression can be improved by not storing term frequencies for each document, because blocks of documents will all have the same term frequency. Within each block, lists can be sorted by document id, to support delta encoding of document ids.

The scoring for this question was 10 points for computational advantages and costs, and only 5 points for compression, because most people had no idea about how well champion lists would compress.

6. PageRank is incorporated into the query likelihood retrieval model as a document prior. Write a formula that shows how PageRank is incorporated into the ranking algorithm. Is the effect of PageRank on the ranking algorithm affected by query length, and is this desirable? Explain and justify your answer. [15 points]

**Answer:**

The query likelihood formula is  $p(q|d) = p(d) \times \prod p(q_i|d)$  (or  $p(q|d) \equiv \log p(d) + \sum \log p(q_i|d)$ ). PageRank is usually incorporated into the prior probability  $p(q|d)$ . The prior probability has the same impact as any individual query term, thus as the query gets longer, the effect of the prior probability is reduced. If PageRank is incorporated into the prior probability, as the query gets longer, it has a smaller effect on the document score. One can argue that this behavior is desired, or undesired. I accepted both answers, as long as the answer was supported by a reasonable argument.

Desired: PageRank is a query-independent, prior probability that a document is relevant. However, as we have more information about how well the query matches the document's contents, the effect of query-independent information such as PageRank should be reduced.

Undesired: PageRank is a query-independent measure of the reliability/popularity/authority of a page. This information is as important as how well the query matches the document's contents, thus the effect of PageRank should not be affected by query length.

10 points for the formula and a coherent explanation of the formula. 5 points for desirable/undesirable.

7. Query log analysis is used to evaluate and tune web search engines. A common requirement is to identify sequences of queries that represent a single information need. Describe an algorithm based on the vector space retrieval model that can automatically identify the boundaries between information needs in a stream of queries from a single individual. For example, the query log below (all from a single individual) should be segmented into 4 information needs, as shown by the dashed lines. Be clear about the strengths and weaknesses of your algorithm. **[15 points]**

ipod.com	2006-04-12	22:28:48	1	<list of top 30 documents>
ipod accessories	2006-04-12	22:29:15	1	<list of top 30 documents>
ipod skins	2006-04-12	22:33:24	1	<list of top 30 documents>
colored ipod skins	2006-04-12	22:39:50	1	<list of top 30 documents>
talking heads	2006-04-12	22:43:01	1	<list of top 30 documents>
heart and achy legs	2006-04-13	20:55:42	3	<list of top 30 documents>
heart attack	2006-04-13	20:56:44	2	<list of top 30 documents>
heart attacks	2006-04-13	20:58:09	1	<list of top 30 documents>
heart health	2006-04-13	22:00:12	1	<list of top 30 documents>
clark shoes	2006-04-13	22:34:06	2	<list of top 30 documents>
saxxon	2006-04-13	22:38:44	3	<list of top 30 documents>
saxxon shoes	2006-04-13	22:39:58	1	<list of top 30 documents>
bernardo shoes	2006-04-13	22:49:08	1	<list of top 30 documents>

**Answer:**

Recall that in the vector-space model, anything can be represented as a vector, and that cosine similarity can be used to determine the similarity of any two vectors. This is the key insight. The algorithm would traverse the log sequentially, comparing query  $i$  to  $i+1$ . If the two are sufficiently similar, it declares them part of the same information need, otherwise it decides that the information need has shifted. (A real algorithm would probably also consider timestamps, but that wasn't necessary for this problem.) There is a possibility of drift, but that probably isn't a serious problem in practice.

The simplest solution is to form the vector for query  $i$  using just the terms in the query. This method is simple, very fast, and probably be sufficient to recognize the similarity among the ipod queries and the heart queries. However, it wouldn't work well for the shoes queries, because there is no vocabulary overlap between "clark shoes" and "saxxon".

A more powerful solution would consider the list of top 30 documents retrieved for each query. Represent query  $i$  by the average (centroid) of the 30 documents it retrieves. Compare the centroid vector for query  $i$  to query  $i+1$ , and call them the same information need if their similarity is above some threshold. This approach considers a broader vocabulary and would be more likely to recognize that "clark shoes" and "saxxon" cover the same topic, but it is also more computationally complex, requires more storage, and might be distracted by irrelevant navigational text on web pages (e.g., "privacy policy", "add to cart").

One could develop more complex algorithms that compensate for interleaved information needs, but that wasn't required for this problem.

## 8. Similarity Metrics

Let  $q = (q_1, \dots, q_n)$  be a query vector,  $x_i = (x_{i,1}, \dots, x_{i,n})$  be document vectors, where  $n$  denotes the size of the vocabulary and  $i = 1, 2, 3, 4$  denote the 4 documents. Term frequencies (TF) are used as the term weighting scheme in these vectors.

- a) Fill in the empty slots in the tables below for vector standardization and similarity computation for documents with respect to the query. See the filled slots as examples. [12 points]

Let the query vector be  $q = (1,2,0)$ , and the size of vocabulary be  $n = 3$ .

Table 1. Vector Standardization (Each cell is 0.4 point; total  $0.4 \times 15 = 6$  points)

Document Vector	Length	$\bar{x}_i$	$\ x_i\ $	Centered $x_i' = x_i - (\bar{x}_i, \bar{x}_i, \bar{x}_i)$	Standard Vector $z_i = (z_{i1}, z_{i2}, z_{i3})$
$x_1 = (1,2,3)$					
$x_2 = (2,4,6)$					
$x_3 = (2,3,4)$					
$x_4 = (1,2,0)$	3	1	$\sqrt{5}$	$x_4' = (0,1,-1)$	$z_4 = \frac{1}{\sqrt{2}}(0,1,-1)$

Table 2. Similarity metrics for document ranking (Each cell is 0.6 point; total 6 points).

Document	$q \cdot x_i$	$\cos(q, x_i)$	PCC $r(q, x_i)$
$x_1 = (1,2,3)$	5		
$x_2 = (2,4,6)$	10		
$x_3 = (2,3,4)$	8		
$x_4 = (1,2,0)$	5		
Doc Ranking	$x_2 > x_3 > \{x_1, x_4\}$		

**Answer:**

- Standardization formulae

$$\bar{x}_i = \frac{1}{n} \sum_{k=1}^n x_{ik}, \quad z_{ik} = \frac{x_{ik} - \bar{x}_i}{\|z\|_2}, \quad \|z_i\| = \sqrt{\sum_{k=1}^n (x_{ik} - \bar{x}_i)^2} \quad (1)$$

- Dot-product

$$q \cdot x_i = \sum_{k=1}^n q_k x_{ik} \quad (2)$$

- Cosine

$$\cos(q, x_i) = \frac{q \cdot x_i}{\|q\| \times \|x_i\|}, \quad \|q\| = \sqrt{\sum_k q_k^2}, \quad \|x_i\| = \sqrt{\sum_k x_{ik}^2} \quad (3)$$

- PCC

$$r(q, x) = z_q \cdot z_x \quad (4)$$

Table 1. Vector Standardization (Each cell is 0.4 point; totally 0.4 x 15 = 6 points)

Document Vector	Total term count	$\bar{x}_i$	$\ x_i\ $	Centered $x_i' = x_i - (\bar{x}_i, \bar{x}_i, \bar{x}_i)$	Standard Vector $z_i = (z_{i1}, z_{i2}, z_{i3})$
$x_1 = (1,2,3)$	6	2	$\sqrt{14}$	$x_1' = (-1,0,1)$	$z_1 = \frac{1}{\sqrt{2}}(-1,0,1)$
$x_2 = (2,4,6)$	12	4	$2\sqrt{14}$	$x_2' = (-2,0,2)$	$z_2 = \frac{1}{\sqrt{2}}(-1,0,1)$
$x_3 = (2,3,4)$	9	3	$\sqrt{29}$	$x_3' = (-1,0,1)$	$z_3 = \frac{1}{\sqrt{2}}(-1,0,1)$
$q = x_4 = (1,2,0)$	3	1	$\sqrt{5}$	$q' = x_4' = (0,1,-1)$	$z_q = z_4 = \frac{1}{\sqrt{2}}(0,1,-1)$

Table 2. Similarity metrics for document ranking (Each cell is 0.6 point; totally 6 points).

Document	$q \cdot x_i$	$\cos(q, x_i)$	PCC $r(q, x_i)$
$x_1 = (1,2,3)$	5	$\frac{5}{\sqrt{5}\sqrt{14}} = \sqrt{\frac{5}{14}} \approx \sqrt{0.36}$	$\frac{-1}{\sqrt{2}\sqrt{2}} = -\frac{1}{2}$
$x_2 = (2,4,6)$	10	$\frac{10}{\sqrt{5}2\sqrt{14}} = \sqrt{\frac{5}{14}} \approx \sqrt{0.36}$	$\frac{-1}{\sqrt{2}\sqrt{2}} = -\frac{1}{2}$
$x_3 = (2,3,4)$	8	$\frac{8}{\sqrt{5}\sqrt{29}} = \sqrt{\frac{64}{145}} \approx \sqrt{0.44}$	$\frac{-1}{\sqrt{2}\sqrt{2}} = -\frac{1}{2}$
$x_4 = (1,2,0)$	5	1	1
Doc Ranking	$x_2 > x_3 > \{x_1, x_4\}$	$x_4 > x_3 > \{x_1, x_2\}$	$x_4 > \{x_1, x_2, x_3\}$

- b) Discuss the major differences of these metrics in ranking documents with respect to a query, such as the tendency to favor longer (or shorter) documents (using the within-document word count as the length measure), the focus on matched proportion instead of absolute term counts, or focus on the variance of term weights from the mean (“center”) of each vector. Overall, which would be the “best” choice for ranking documents in your opinion? Justify your answer. [13 points]

**Answer:**

Dot-product tends to favor longer (many words) or more verbose (high TF) documents. Cosine, on the other hand, would favor shorter documents among those with the same dot-product values. In other words, cosine similarity focuses on the proportion of matching terms instead of absolute counts. PCC is the only metric (among those three) focusing on variances from the vector-specific means. For example,  $x_1$ ,  $x_2$  and  $x_3$  have different lengths and different means, but the same standard vector. Thus using PCC to rank documents, these three documents are indistinguishable. Cosine or dot-product are more sensible choice than PCC for ranking documents against a query because we want to find relevant documents which typically have more shared words with the query than irrelevant documents. PCC, on the other hand, focuses on local rescaling of term weights in the standardization. It makes sense to remove users' biases in choosing terminology, but also has the effect of de-emphasizing matching terms among queries and documents, making retrieval less effective.