

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

**Midterm Exam**  
Information Retrieval and Web Mining (15-493)  
October 13, 2009

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. How many terms occur more than 20,000 times in a 1,000,000,000 word corpus? Show your work. **[6 points]**

**Answer:**

Zipf's Law says that Rank  $\times$  Frequency = Constant. The constant is approximately  $0.1 \times$  Corpus Size.

$$\text{Rank} \times 20,000 = 1,000,000,000 / 10$$

$$\text{Rank} = 1,000,000,000 / 200,000$$

$$= 10,000 / 2$$

$$= 5,000$$

**Grading note:**

Some people pointed out that I said 'more than' instead of 'at least', thus the answer is really 4,999. I intended 'at least', but the wording was unclear, thus I accepted both answers.

3 points for knowing that Zipf's Law should be used.

1 point deducted for simple math errors (it was surprising how many people think  $10,000/2 = 10,000$ ).

1-3 points deducted for irrelevant information or errors in applying Zipf's Law correctly.

2. As you know, there are many metrics for evaluating a single ranked list. For each of the following search scenarios, suggest the most appropriate evaluation metric. Justify your choice.

- a. A prospective student searching for the CMU homepage. **[6 points]**

**Answer:**

Reciprocal Rank, because there is just a single answer, and we want a stiff penalty for when the home page isn't ranked first.

- b. A lawyer searching a set of documents to find all of those that pertain to her case. She bills her clients by the hour, and she is evaluated primarily by whether she wins the case, so she does not mind sifting through all of the documents that were returned. **[6 points]**

**Answer:**

(Set-based) Recall. This lawyer will look at everything, so it doesn't really matter how the returned documents are ordered. Instead, the key metric is how many relevant documents are in the result set.

- c. An American football fan doing some research on the web about the SuperBowl. This is a recreational activity, done for fun. Some search results will provide a lot of information (the latest news and rankings, upcoming matches, the history of the SuperBowl, ...), while other results will be less interesting or irrelevant. **[6 points]**

**Answer:**

NDCG. There is a wide variety of information about the SuperBowl on the web. Some of it is very relevant and interesting to our SuperBowl fan. Some of it is less relevant and interesting, so we want a metric that uses multi-valued relevance assessments.

3. Describe the main processes involved in creating an inverted index on a standard desktop computer. Assume that documents are provided in a canonical format, one-at-a-time. (In other words, you can ignore where documents come from, and what format they are in.) Describe how documents are transformed into inverted lists. This is running on a desktop computer with other processes, so be clear about how memory is used, and what is stored on disk. [15 points]

**Answer:**

The parser/indexer received a document in a canonical format. There are essentially four steps to creating an inverted index. (See lecture 4, slide 29.)

- a. A block of memory is created to store fragments of inverted lists. It is initially empty.
- b. As each document is received, the parser/indexer converts the document to canonical tokens, for example, by converting each token to lowercase, discarding stopwords, stemming the token, and doing other forms of lexical processing. When lexical processing of the token is complete, the in-memory inverted list for the token is updated with frequency and position information.
- c. Eventually, after parsing/indexing many documents, the block of memory used to store fragments of inverted lists becomes full. It is written to disk, and then reinitialized (i.e., contains no inverted list fragments). Note that the block on disk contains all of the inverted list information for a sequence of documents, e.g., documents 10,001 to 20,000.
- d. After all documents have been parsed and indexed, the blocks of inverted list fragments are examined in parallel. For each term (e.g., 'apple'), the inverted list fragments from each block are read, concatenated together to produce a complete inverted list, and written to the final index (a file of complete inverted lists).

**Grading notes:**

Steps b, c, and d (above) were each worth 5 points.

Many answers suggested that when inverted list fragments are flushed to disk, they are merged with the inverted lists already stored on disk. This architecture is possible, and is probably done in some systems. However, it requires a sophisticated solution to manage inverted lists on disk, otherwise the I/O costs are very high. (What if there is no room to append the new data at the end of the inverted list on disk? The inverted list needs to be copied somewhere else on disk. Imagine doing this for many lists each time you flush data to disk.) No credit for step d unless this issue was addressed in some way.

Points were deducted for assuming that indexing could be done entirely in memory.

4. Describe 4 text representations (e.g., Title) that might be reasonable in a web search system, and the advantages of each. Show how the language modeling approach to information retrieval can integrate multiple text representations in the retrieval model. Show the formulas. Be clear about what each term means. [15 points]

**Answer:**

We often discussed 4 representations for web search engines: Title, Inlink, Body, and URL text.

Advantages:

- The title text is usually a concise and accurate description of a web page. There are few distracting or noisy terms.
- The inlink text is usually a concise description of a web page. Because it is usually assigned by others (i.e., not the author), it is more likely to be objective (i.e., not spammed), and it may be more likely to describe the page as others see it.
- The body text is comprehensive and detailed, but it is also easy to spam.
- The URL text is concise and may contain terms that are important in navigational queries. For example, 'cmu' appears in the URL of every page from cmu.edu, but the word 'cmu' may not appear on many of those pages. For non-text data, the URL may provide clues about the image contents, e.g., worlds-strongest-dog.jpg (a real filename!).

The language modeling approach to information retrieval can integrate multiple text representations as follows:

$$p(q|d) = p(d) \prod_i p(q_i|d)$$

$$p(q_i|d) = \sum_j \lambda_j p(q_i|d_j)$$

$$\sum_j \lambda_j = 1$$

where  $q$  is the query,  $q_i$  is the  $i$ 'th query term,  $d$  is the document language model,  $d_j$  is the language model for the  $j$ 'th representation of the document (e.g., title, inlink, body, or url), and  $\lambda_j$  is the weight of the  $j$ 'th representation.  $p(q|d)$  is the probability of query  $q$  given the language model for document  $d$ .  $p(d)$  is the prior probability of the document.  $p(q_i|d_j)$  would probably be based on a maximum likelihood estimate (MLE) that is smoothed, e.g., with Jelinick-Mercer or Bayesian smoothing with Dirichlet priors, but it isn't necessary to show smoothing for this answer.

**Grading notes:**

Each of the three answer components was worth 5 points.

I accepted other types of representations, e.g., other HTML fields, text annotations, etc, as long as they were justified appropriately.

I also accepted models similar to what we discussed for XML retrieval.

Many answers didn't provide 4 representations, or didn't describe the advantages of the representations they listed. Read the instructions!

Some answers confused text representations and document priors. I didn't accept document priors.

5. Language modeling

- a. What are 2 benefits of using smoothing in language modeling approaches to information retrieval? **[8 points]**

**Answer:**

A language model estimated from a small sample (e.g., a single document) produces unreliable probability estimates, for example, overestimating the probability of terms that occur, and underestimating the probability of terms that are not observed. Smoothing ‘borrows’ probability mass from terms that are observed in the document, and provides non-zero probabilities for terms that were not observed in the document (soft-match, or probabilistic AND), thus improving the probability estimates for all terms. Smoothing with a corpus language model also provides an expectation about how often a term should occur, which reduces the impact of terms that occur frequently in the corpus (the ‘idf-like’ effect).

- b. Describe one of the smoothing methods studied in class (with formula). Which of the benefits described above does it provide, and how? **[7 points]**

**Answer:**

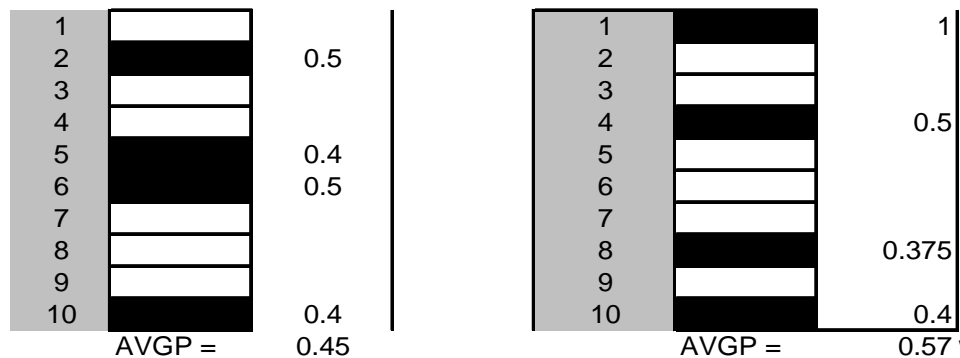
The main benefit of Bayesian smoothing with Dirichlet priors is thought to be that it improves the document language model.

The main benefit of Jelenick-Mercer smoothing is thought to be that it provides the ‘idf-like’ effect.

6. Evaluating ranked lists

Given a query, systems A and B returned a ranked list of 10 documents, respectively, as listed in the table below, where the black boxes are relevant documents and the white boxes are irrelevant documents. Assume that the total number of relevant documents in four.

Compute the Average Precision (AP) for both systems, using the column “Prec @ i” to show the intermediate results of the calculation for systems A and B, respectively. **[9 points]**



## 7. Hypertext Retrieval

PageRank scores of hyperlink documents can be computed in the following steps:

- Line 1.  $\vec{x}^{(0)} = \frac{1}{n} \vec{1} = \left( \frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n} \right)^T$
- Line 2. for  $k = 1, 2, \dots, K$  compute:
- Line 3.  $\vec{x}^{(k)} = B\vec{x}^{(k-1)}$
- Line 4. return  $\vec{x}^{(K)}$  as the vector of the PageRank scores.

### Questions

- a. Define matrix B in line 3 for the PageRank method. [10 points]

**Answer:** Let us define the  $n \times n$  matrices  $M$ ,  $U$  and  $B$  as below:

$$M[i, j] = \frac{I(i \rightarrow j)}{\sum_{j'=1}^n I(i \rightarrow j')} \text{ for } i, j = 1, \dots, n;$$

$I(i \rightarrow j) \in \{0, 1\}$  indicates whether or not page  $i$  has a link to page  $j$ ;

$$U[i, j] = \frac{1}{n} \text{ for } i, j = 1, \dots, n.$$

$$B = \alpha U + (1 - \alpha)M, \quad 0 < \alpha \leq 1.$$

- b. Use the eigen system of matrix B to prove the convergence of  $\vec{x}^{(K)}$ . [6 points]

**Answer:** To how  $\vec{x}^{(K)}$  converges to the principal eigenvector of matrix B, we have:

$$\begin{aligned} \vec{x}^{(K)} &= B\vec{x}^{(K-1)} = B^K \vec{x}^{(0)} \\ &= \underbrace{U\Lambda U^T}_B \underbrace{U\Lambda U^T}_B \dots \underbrace{U\Lambda U^T}_B \vec{x}^{(0)} = U\Lambda^K \underbrace{U^T \vec{x}^{(0)}}_{\vec{c}} \\ \vec{c} \equiv U^T \vec{x}^{(0)} &= \begin{pmatrix} \vec{u}_1^T \vec{x}^{(0)} \\ \vec{u}_2^T \vec{x}^{(0)} \\ \vdots \\ \vec{u}_n^T \vec{x}^{(0)} \end{pmatrix} \equiv \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} \\ \vec{x}^{(K)} = U\Lambda^K \vec{c} &= (\lambda_1 \vec{u}_1, \lambda_2 \vec{u}_2, \dots, \lambda_n \vec{u}_n) \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} = \sum_{k=1}^K c_k \lambda_k^K \vec{u}_k \\ &\approx c_1 \lambda_1^K \vec{u}_1 \quad (\lambda_1^K \gg \lambda_{i1}^K \text{ for a sufficiently large } K) \end{aligned}$$

Notice that the resulting vector is proportional to the first eigenvector of B, not necessarily identical. Regardless, the same constant ( $c_1 \lambda_1$ ) is multiplied to the score of each page in  $\vec{u}_1$ , meaning that the constant does not change the relative ranking of pages.

- c. Define the Topic Sensitive PageRank (TSPR) method by modifying matrix B and formula in line 3, and provide the formula for calculating the TSPR scores of pages at the query time. [6 points]

**Answer:** Let  $j = 1, 2, \dots, m$  be the topic index ( $m \leq n$ ) and  $n_j$  be the number of pages in topic  $j$ , and define the topic-specific teleportation vector as:

$$\vec{r}_{ji}^{(0)} = \begin{cases} \frac{1}{n_j} & \text{if page } i \text{ belongs topic } j \\ 0 & \text{otherwise} \end{cases}$$

The TSPR for each topic is calculated as:

$$\vec{r}_j^{(k)} = (1 - \alpha) M^T \vec{r}_j^{(k-1)} + \alpha \vec{r}_j^{(0)}$$

The online scoring given a query is calculated as:

$$\vec{r}_q^{(TSPR)} = \sum_{j=1}^m \Pr(j | q) \vec{r}_j$$

where  $\Pr(j | q) \propto \Pr(j) \Pr(q | j)$  can be estimated using NaiveBayes statistical classifiers.