CS54701 Spring 2016 Assignment 2 Sample Solution

Compiled from Kexin Pei, Kai Tang and Qing Wei

1 Boolean Retrieval (10 minutes, 4 points)

Give an example of an information need where features from Boolean Retrieval would be useful. That is, a standard vector-space model approach such as TF-IDF weighting and cosine similarity would not give the desired results, but using some form of boolean query would likely give substantially better results. Describe what specific boolean query capabilities/features are needed.

I would suggest you give a non-boolean query example and a boolean query example, and use this to explain why the boolean would do a better job for the particular information need.

If we want to determine the plays of Shakespeare that contain the words 'Brutus' AND 'Caesar' AND NOT 'Calpurnia'. For this kind of query, Boolean retrieval will give exact information that we needed. However, standard vector-space model cannot handle NOT 'Calpurnia'.

2 TF-IDF weighting (10 minutes, 4 points)

Given the following document term frequencies:

	DocA	*DocB*	*DocC*
CS54701	3	0	1
Final	2	0	4
May	1	5	2

Compute the TFIDF (ltc with natural logarithm) weight for *DocA only.*

A. Give the method you are using to determine weights. You may do this by giving the formula used.

$$l : 1 + \ln(\mathrm{tf}_{t,d})$$

$$t : \ln \frac{N}{\mathrm{df}_i}$$

$$c : \frac{1}{\sqrt{\sum_{i=1}^{M} w_i^2}}$$

B. Complete the table for *DocA* only.

N=3, $\mathrm{df}_1 = 2, \mathrm{df}_2 = 2, \mathrm{df}_3 = 3$; $\mathrm{tf}_{1,\mathrm{docA}} = 3, \mathrm{tf}_{2,\mathrm{docA}} = 2, \mathrm{tf}_{3,\mathrm{docA}} = 1$

$1 + \ln(3) = 2.098612$, $1+\ln(2) = 1.693147$, $1+\ln(1)=1$

$\ln \frac{3}{2} = 0.4054651; \ln \frac{3}{3} = 0$

	tf-raw	tf-weigth	df	idf	tf.idf	normalized
CS54701	3	2.098612	2	0.4054651	0.8509139	0.7782829
Final	2	1.693147	2	0.4054651	0.686512	0.6279138
May	1	1	3	0	0	0

	DocA	*DocB*	*DocC*
CS54701	0.7782829	x	x
Final	0.6279138	x	x
May	0	x	x

3 Language Models (15 minutes, 9 points)

Given the following document term frequencies:

	DocA	*DocB*	*DocC*
CS54701	3	0	1
Final	2	0	4
May	1	5	2

Compute a generative unigram language model for *DocC only.*

A. Complete the table for *DocC* only.

	DocA	*DocB*	*DocC*
CS54701	x	x	$\frac{1}{7}$
Final	x	x	$\frac{4}{7}$
May	x	x	$\frac{2}{7}$

B. Explain how you would use this language model to rank the documents in response to a query. The Language Modeling approach attempts to model the query generation process: Documents are ranked by the probability that a query would be observed as a random sample from the respective document model.

The Language Modeling approach attempts to model the query generation process: Documents are ranked by the probability that a query would be observed as a random sample from the respective document model.

For example, if the query is q = "CS54701 Final".

$$\begin{aligned}
\hat{P}(q|M_d) &= \prod_{t\in q} \hat{P}_{\text{MLE}}(t|M_d) \\
&= \prod_{t\in q} \frac{\text{tf}_{t,d}}{L_d} \\
\hat{P}(q|\,\text{DocA}) &= \frac{1}{2} \times \frac{1}{3} \\
&= \frac{1}{6} \\
&\approx 0.167 \\
\hat{P}(q|\,\text{DocB}) &= 0 \times 0 \\
&= 0 \\
\hat{P}(q|\,\text{DocC}) &= \frac{1}{7} \times \frac{4}{7} \\
&= \frac{4}{49} \\
&\approx 0.082
\end{aligned}$$

$$\hat{P}(q|\,\text{DocA}) > \hat{P}(q|\,\text{DocC}) > \hat{P}(q|\,\text{DocB})$$

So the rank is $DocA > DocC > DocB$.

C. Is the following statement true or false? Explain your answer: A probabilistic retrieval model gives each document a score that is the probability that the document meets the user's information need.

False. In language model, the output is not the probability that the document meets the user's information need, but the probability that the certain document can generate this query, given the language model of a document.

4 Latent Semantic Indexing (15 minutes, 5 points)

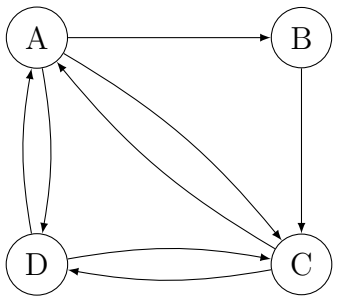
One purported advantage of Latent Semantic Indexing is that it can retrieve documents that address the same concept as a query, even if the words do not match. Explain briefly how this can happen. While you may find it useful to give an example matrix, I do not expect you to do an actual singular-value decomposition; an example that shows how this could be happen is fine, even if the matrices don't actually multiply out to something close to the original term-document matrix.

Section 3.2 in `http://www.engr.uvic.ca/~seng474/svd.pdf` explains this geometrically.

I didn't expect you to write all of this to get full points. A concise explanation on how LSI uses SVD to discover latent topics in the term-document matrix with a simple example was needed to get full points.

5 Link Analysis/PageRank (15 minutes, 6 points)

A. Construct the stochastic matrix for the following graph:



The adjacent matrix M is given as follows

$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

So the stochastic matrix B will be:

$$\begin{pmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 1 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \end{pmatrix}$$

B. Show the first three iterations of the eigenvector computation.

The first three iterations of the principal eigenvector computation using the power iteration method is as follows: Let the initial state $\vec{x_0}$ =(1 0 0 0)

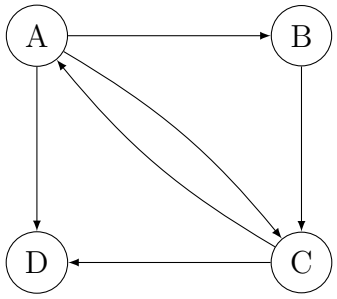
$$\vec{x_1} = \vec{x_0}B = (0, \frac{1}{3}, \frac{1}{3}, \frac{1}{3})$$

$$\vec{x_2} = \vec{x_1}B = (\frac{1}{3}, 0, \frac{1}{2}, \frac{1}{6})$$

$$\vec{x_3} = \vec{x_2}B = (\frac{1}{3}, \frac{1}{9}, \frac{7}{36}, \frac{13}{36})$$

C. Consider a modified graph in which the edges (d, a) and (d, c) are removed. What is the problem with this graph and how can it be fixed?

If we remove edges (d, a) and (d, c), then node D will become a sink and the power iteration method converges to the 0 vector.



To fix this, we can add the teleport operation[1]. In the teleport operation the surfer jumps from a node to any other node in the web graph. This could happen because he types an address into the URL bar of his browser. The destination of a teleport operation is modeled as being chosen uniformly at random from all web pages. If there are N nodes in a graph, the teleport operation takes the surfer to each node with probability 1/N. The surfer would also teleport to his present position with probability 1/N.

We can use the teleport operation in two ways: (1) When at a node with no out-links, the surfer invokes the teleport operation. (2) At any node that has outgoing links, the surfer invokes the teleport operation with probability $0 < \alpha < 1$ and the standard random walk with probability 1 - α, where α is a fixed parameter chosen in advance.

[1] http://nlp.stanford.edu/IR-book/html/htmledition/pagerank-1.html