October 20, 2016

CSCI 2820
Application 2, part 1: Building a Search Engine

Due Thursday, October 27

In this first part of Application 2, you will study an application of vector spaces and column spaces. In part 2, spanning sets, rank, and bases will be important.

For this assignment, you'll begin to build a search engine that identifies the documents in a database related to a user's query. The search engine will be based on the vector space model described below. It is rumored that this model has been used successfully in some commercial search engines, and it has been used in a variety of other smaller scale ventures such as an automatic grading program [6].

The Vector Space Model

In the vector space model of information retrieval (IR), a vector is used to represent each item or *document* in a collection. Each component of the vector reflects a particular concept, keyword, or *term* associated with the given document.

Some Preliminaries

A database containing a total of d documents described by t terms is represented as a $t \times d$ *term-by-document matrix* A. The d vectors representing the d documents form the columns of the matrix. Thus, the matrix element a_{ij} is the frequency at which term i occurs in document j [2]. In the parlance of the vector space model, the columns of A are the *document vectors*, and the rows of A are the *term vectors*. The semantic content of the database is wholly contained in the column space of A, meaning that the document vectors span that content. Not every vector represented in the column space of A has a specific interpretation in terms of the document collection itself (i.e., a linear combination of vectors corresponding to two document titles may not translate directly into a meaningful document title.) What is important from an information retrieval perspective, however, is that we can exploit geometric relationships between document vectors to model similarities and differences in content. We can also compare term vectors geometrically in order to identify similarities and differences in term usage.

For text collections spanning many contexts (e.g., an encyclopedia), the number of terms is often much greater than the number of documents: $t \gg d$. In the case of the Internet, the situation is reversed. At this time, the term-by-document matrix using the content of the largest English language dictionary as terms and the set of all web pages as documents was reported to be about $300,000 \times 30,000,000,000$ [1, 4, 11, 8, 9]. The present size of Google's index is at least 30 billion webpages [9]. As a document generally uses only a small subset of the entire dictionary of terms generated for a given database, most of the elements of a term-by-document matrix are zero.

In a vector space IR scheme, a user queries the database to find relevant documents, somehow using the vector space representation of those documents. The query is a set of terms represented just like a document. Again, it is likely that many of the terms in the database do not appear in the query, meaning that many of the query vector components are zero. Query matching is finding the documents most similar to the query in use of terms. In the vector space model, the documents selected are those geometrically closest to the query according to some measure.

The Similarity Measure

One common measure of similarity is the cosine of the angle between the query and document vectors. If the term-by-document matrix A has columns $a_j, j = 1, \ldots, d$, those d cosines are computed according to the formula

$$\cos\theta_j = \frac{a_j^T q}{\| a_j \|_2 \| q \|_2} = \frac{\sum_{i=1}^t a_{ij} q_i}{\sqrt{\sum_{i=1}^t a_{ij}^2}\sqrt{\sum_{i=1}^t q_i^2}}, \tag{1}$$

for $j = 1, \ldots, d$, where the Euclidean vector norm $\| x \|_2$ is defined by $\| x \|_2 = \sqrt{x^T x} = \sum_{i=1}^t x_i^2$ for any real t-dimensional vector x. Because the query and document vectors are typically sparse, the dot product and norms in Equation (1) are generally inexpensive to compute. Furthermore, the document vector norms $\| a_j \|_2$ need be computed only once for any given term-by-document matrix. Note that multiplying either a_j or q by a constant does not change the cosine value. Thus, we may scale the document vectors or queries by any convenient value.

An Example

Figure 1 demonstrates how a simple collection of five titles described by six terms leads to a 6×5 term-by-document matrix. Because the content of a document is determined by the relative frequencies of the terms and not by the total number of times particular terms appear, the matrix elements in this example are scaled so that the Euclidean norm of each column is one. That is, $\| a_j \|_2 = 1$ for columns a_j, $j = 1, \ldots, 5$.

The choice of terms used to describe the database determines not only its size but also its utility. In our example, we used only the terms directly related to cooking meaning that the reader interested in *French cooking* in particular would have no way of retrieving relevant documents. In this case, adding the terms **French** and **Viennese** to describe the nationalities covered would broaden the representation of the database semantics in a helpful way. On the other hand, including very common terms like **to** or **the** would do little to improve the quality of the term-by-document matrix. The process of excluding such high frequency words is known as *stoplisting* [7].

In constructing a term-by-document matrix, terms are usually identified by their word stems [10]. In our example, the word *pastries* counts as the term **pastry**, and the word *baking* counts as the term **bake**. The use of *stemming* in information retrieval dates back to the 1960s [12]. Stemming reduces storage requirements by decreasing the number of words maintained [13].

Query Matching

Using the small collection of titles from Figure 1, we can illustrate query matching based on angles in a 6-dimensional vector space. Suppose that a user in search of cooking information initiates a search for books about *baking bread*. The corresponding query is written as the vector

$$q^{(1)} = (\,1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0\,)^T$$

with nonzero entries for the terms **bake** and **bread**. The search for relevant documents is carried out by computing the cosines of the angles θ_j between the query vector $q^{(1)}$ and the document vectors a_j by Equation (1). A document is returned as relevant only if the cosine of the angle it makes with the query vector is greater than some threshold or cutoff value. A practical implementation might use a stringent cutoff like 0.9 [2], but for our small example we use a cosine threshold of 0.5.

For the query $q^{(1)}$, the only nonzero cosines are $\cos\theta_1 = 0.8165$ and $\cos\theta_4 = 0.5774$. Hence, all of the documents concerning baking bread (the first and fourth) are returned as relevant. The second, third and fifth documents, which concern neither of these topics, are correctly ignored.

If the user had simply requested books about *baking*, however, the results would have been markedly different. In this case, the query vector is given by

$$q^{(2)} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}^T,$$

and the cosines of the angles between the query and five document vectors are, in order, 0.5774, 0, 0, 0.4082, and 0. Only the first document, a book about baking bread, makes the cosine cutoff. The fourth document, which is in fact a more comprehensive reference about baking, is not returned as relevant.

The IR community has developed a variety of approaches to respond to such failures of the basic vector space model. Those techniques typically affect how the data are represented in the term-by-document matrix. Examples include term weighting schemes [5, 14], use of a *controlled vocabulary* (a specified set of allowed terms [10]), and replacing the exact term-by-document matrix by a low-rank approximation to that matrix [3]. The latter is a technique of removing extraneous information or *noise* from the database representation. In the second part of this application problem, we will examine the effects of rank reduction.

The $t = 6$ **terms**:

T1:	bak(e,ing)
T2:	recipes
T3:	bread
T4:	cake
T5:	pastr(y,ies)
T6:	pie

The $d = 5$ **document** titles:

D1:	How to Bake Bread Without Recipes
D2:	The Classic Art of Viennese Pastry
D3:	Numerical Recipes: The Art of Scientific Computing
D4:	Breads, Pastries, Pies and Cakes : Quantity Baking Recipes
D5:	Pastry: A Book of Best French Recipes

The 6×5 term-by-document matrix before normalization, where the element $\hat{a}_{ij}$ is the number of times term i appears in document title j:

$$\hat{A} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

The 6×5 term-by-document matrix with unit columns:

$$A = \begin{pmatrix} 0.5774 & 0 & 0 & 0.4082 & 0 \\ 0.5774 & 0 & 1.0000 & 0.4082 & 0.7071 \\ 0.5774 & 0 & 0 & 0.4082 & 0 \\ 0 & 0 & 0 & 0.4082 & 0 \\ 0 & 1.0000 & 0 & 0.4082 & 0.7071 \\ 0 & 0 & 0 & 0.4082 & 0 \end{pmatrix}$$

Figure 1: The construction of a term-by-document matrix A.

The Assignment

1. Read and understand all of the above text.

2. Write a search engine script in MATLAB. Your script should take as inputs a term-document matrix A, a query vector, and a tolerance. The output of your script must repeat the query and must clearly identify which documents (columns of A) are relevant to that query. It must also show the cosine value for each document deemed relevant.

3. Check that you can reproduce all of the results in the above text using your search engine script. The queries you must try are

$$\begin{aligned} q^{(1)} &= (1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0)^T \\ q^{(2)} &= (1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)^T \end{aligned}$$

4. Experiment with the following two queries:

   (a) What is the difference between pastry and bread?

   (b) Show me all recipes for pastry

   Comment on the correctness of the vector space model for these queries. What is the best choice of cosine cutoff for these and the above queries?

5. Turn in a listing of your script and the results of all of your tests including your comments from 4. (20 points).

6. We'll complete the second part after studying the QR factorization, so hold onto your MATLAB script! (You may also use the one I will provide in the solutions to this part.)

You may write the programs in this assignment with one partner. Partners may turn in a single paper with both names on it. Please put both names together at the top of the paper.

Bibliography

[1] D. Berg, *A Guide to the Oxford English Dictionary*, Oxford University Press, Oxford, 1993.

[2] M. Berry, S. Dumais, and G. O'Brien, *Using linear algebra for intelligent information retrieval*, SIAM Review, 37 (1995), pp. 573–595.

[3] M. W. Berry, Z. Drmač, and E. R. Jessup, *Matrices, vector spaces, and information retrieval*, SIAM Review, 41 (1999), pp. 335–362.

[4] K. Bharat and A. Broder, *Estimating the relative size and overlap of public web search engines*, in 7th International World Wide Web Conference, Elsevier Science, 1998. Paper FP37.

[5] S. Dumais, *Improving the retrieval of information from external sources*, Behavior Research Methods, Instruments, & Computers, 23 (1991), pp. 229–236.

[6] P. Foltz, R. Laham, T. Landauer, W. Kintsch, and B. Rehder, *Methods for analysis and evaluation of the semantic content of writing*. patent pending, 1997.

[7] W. Frakes and R. Baeza-Yates, *Information Retrieval: Data Structures & Algorithms*, Prentice-Hall, Englewood Cliffs, NJ, 1992.

[8] *Google offers immediate access to 3 billion web documents*. http://www.google.com/press/pressrel/3billion.html, December 11, 2001.

[9] *The size of the world wide web (the internet)*. http://www.worldwidewebsize.com/, November 6, 2013.

[10] G. Kowalski, *Information Retrieval Systems: Theory and Implementation*, Kluwer Academic Publishers, Boston, 1997.

[11] S. Lawrence and C. Giles, *Searching the world wide web*, Science, 280 (1998), pp. 98–100.

[12] J. Lovins, *Development of a stemming algorithm*, Mechanical Translation and Computational Linguistics, 11 (1968), pp. 22–31.

[13] G. Salton, *Automatic Information Organization and Retrieval*, McGraw Hill, New York, 1968.

[14] K. Sparck Jones, *A statistical interpretation of term specificity and its applications in retrieval*, Journal of Documentation, 28 (1972), pp. 11–21.