Vector Space Calculations without Linear Algebra

Abstract – This is an introductory tutorial for those interested in vector space models, but that lack of a linear algebra background. The calculations can be easily replicated with a spreadsheet, online calculator, or by hand.

Keywords: vector space model, linear algebra, term-document, term-query matrices, dot products, cosine similarities, local weights, global weights
Published: 03-04-2016; Updated: 03-19-2016
© E. Garcia, PhD; admin@minerazzi.com

Introduction

In a previous tutorial (Garcia, 2016a), we presented a linear algebra approach for some vector space models used in IR (Baeza-Yates & Ribeiro-Neto, 1999; Rijsbergen, 2004; Grossman & Frieder, 2004).

This time we present a simpler approach that does not require of linear algebra. The calculations can be done with a spreadsheet, calculator, or by hand. So this tutorial is suitable for those interested in learning about vector space models, but that lack of a linear algebra background.

Documents and Queries as Points

Let's represent a document, d_j, and query, q, as points in a two-dimensional space where each dimension corresponds to a separate term. We call this a *term space*. The Cartesian coordinates of each point in this space are weights computed with a weighting scheme.

Assume that $d_j(w_{1,j}, w_{2,j})$ and $q(w_{1,q}, w_{2,q})$ are the coordinates of d_j and q, and that there is a third point, o, with coordinates $o(0, 0)$; i.e. at the origin of the term space. See Figure 1.

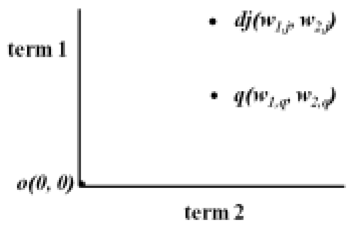

Figure 1. Document and query points in a two-dimensional term space.

Multiplying coordinates and taking summation yields a quantity called the *dot product*

$$\textit{dot product} = w_{1,j} * w_{1,q} + w_{2,j} * w_{2,q} \tag{1}$$

If more than two terms (dimensions) are used to define the term space, we simply compute the dot product by adding the corresponding products to (1). Thus, for *n* number of unique terms (1) can be written as

$$\textit{dot product} = \sum_{i=1}^{n} w_{i,j}\, w_{i,q} \tag{2}$$

Now to visualize how far $d_j(w_{1,j}, w_{2,j})$ and $d_j(w_{1,q}, w_{2,q})$ are from $o(0, 0)$, we can draw straight lines between these and proceed as follows:

- Take coordinate differences and square them.
- Add all squared differences and square root the result.

So we end up with the *Euclidean distance*, L, between the points. See Figure 2.

$$\mathrm{L}_{\mathrm{dj,0}} = \sqrt{\left(w_{1,j} - 0\right)^2 + \left(w_{2,j} - 0\right)^2} = \sqrt{{w_{1,j}}^2 + {w_{2,j}}^2} = \sqrt{\textstyle\sum_{i=1}^{n} w_{i,j}^2} \tag{3}$$

$$\mathrm{L}_{\mathrm{q,0}} = \sqrt{\left(w_{1,q} - 0\right)^2 + \left(w_{2,q} - 0\right)^2} = \sqrt{{w_{1,q}}^2 + {w_{2,q}}^2} = \sqrt{\textstyle\sum_{i=1}^{n} w_{i,q}^2} \tag{4}$$

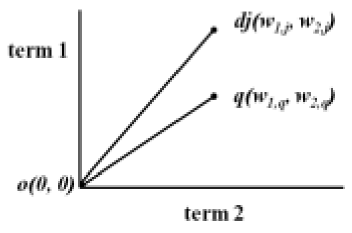

Figure 2. Document and query Euclidean distances from the origin.

Documents and Queries as Vectors

The straight lines shown in Figure 2 can be replaced by vectors. A vector is a quantity with *direction* and *magnitude*. The direction of a vector is given by the angle formed relative to its dimensions; its magnitude is an absolute value given by its length relative to the space origin, also known as the L_2-norm. Figure 3 shows d_j and q as the vectors $\mathbf{d_j}$ and $\mathbf{q}$, often written as $\overrightarrow{\mathbf{d_j}}$ and $\vec{\mathbf{q}}$.

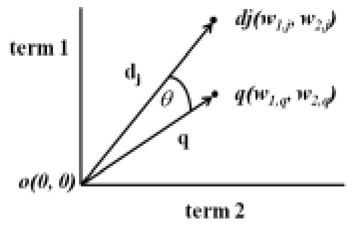

Figure 3. Document and query vectors in a term space.

In Figure 3, as the two vectors get closer the cosine of the angle between vectors, cos(θ), increases, approaching 1. If the vectors are superimposed, cos(θ) = 1, and the two vectors are completely similar to one another. So cos(θ) is a measure of the similarity between the $\mathbf{d_j}$ and $\mathbf{q}$ vectors that we may call the *cosine similarity*, $sim(d_j, q)$

$$sim(d_j, q) = \frac{\mathbf{d_j} \bullet \mathbf{q}}{\|\mathbf{d_j}\| \, \|\mathbf{q}\|} = \frac{\sum_{i=1}^{n} w_{i,j} \, w_{i,q}}{\sqrt{\sum_{i=1}^{n} w_{i,j}^2} \sqrt{\sum_{i=1}^{n} w_{i,q}^2}} \tag{5}$$

where • means, you guessed right, the dot product of $\mathbf{d_j}$ and $\mathbf{q}$. Expressions (1) to (5) show that $sim(d_j, q)$ is a vector dot product normalized by vector magnitudes where

$\mathbf{d_j} \bullet \mathbf{q} = \sum_{i=1}^{n} w_{i,j} w_{i,q}$ = dot product

$\|\mathbf{d_j}\| \, \|\mathbf{q}\| = \sqrt{\sum_{i=1}^{n} w_{i,j}^2} \sqrt{\sum_{i=1}^{n} w_{i,q}^2}$ = magnitudes product

So we can use (5) to rank D documents against a query in decreasing order of cosine similarities. Let's illustrate this with the example given in our previous tutorial (Garcia 2016a).

Problem

A collection of five "documents" ($D = 5$) is searched with the query, *q*, *latent semantic indexing.*

d_1 = LSI tutorials and fast tracks.

d_2 = Books on semantic analysis.

d_3 = Learning latent semantic indexing.

d_4 = Advances in structures and advances in indexing.

d_5 = Analysis of latent structures.

Document terms are not reduced to *word roots*. However, the documents are

1. **linearized**, by removing markup tags, comments, style instructions, and scripts.
2. **tokenized**, by removing punctuation and lowercasing terms.
3. **filtered**, by removing stopwords; i.e., frequently used, or redundant words like *and, of, in…*

Survival terms are arranged as an index of terms with frequency data. See Table 1.

Table 1. Index terms frequency data.

Index terms	***q***		d_1	d_2	d_3	d_4	d_5
advances	0		0	0	0	2	0
analysis	0		0	1	0	0	1
books	0		0	1	0	0	0
fast	0		1	0	0	0	0
indexing	1		0	0	1	1	0
latent	1		0	0	1	0	1
learning	0		0	0	1	0	0
lsi	0		1	0	0	0	0
semantic	1		0	1	1	0	0
structures	0		0	0	0	1	1
tracks	0		1	0	0	0	0
tutorials	0		1	0	0	0	0

Depending on the nature of documents and queries, index terms can be weighted with a given weighting scheme or a combination of these (Chisholm & Kolda, 1999). For instance, Salton tried a total of 1800 combinations of which 287 were found to be distinct (Salton & Buckley, 1987).

To start familiarizing yourself with two of the several models out there, in this tutorial query terms are scored with the Term Count Model and document terms with the TF-IDF Model.

Query: Term Count Model (FREQ Model): $w_{i,q} = L_{i,q} = f_{i,q}$

- $w_{i,q}$ = weight of index term i in query q.
- $L_{i,q}$ = local weight defined as the frequency of index term i in query q.

Documents: TF-IDF Model: $w_{i,j} = L_{i,j}\, G_i = f_{i,j}\, log(D/d_i)$

- $w_{i,j}$ = weight of index term i in document j.
- $L_{i,j}$ = local weight defined as the frequency of index term i in document j.
- $G_i = log(D/d_i)$ = global weight, defined as *Inverse Document Frequency (IDF)* where D is the collection size and d_i the number of documents that mention index term i.

Solution

Table 2 shows the result of weighting index terms with these models.

Table 2. Index term weights.

Index terms	$w_{i,q}$		$w_{i,1}$	$w_{i,2}$	$w_{i,3}$	$w_{i,4}$	$w_{i,5}$
advances	0		0.00	0.00	0.00	1.40	0.00
analysis	0		0.00	0.40	0.00	0.00	0.40
books	0		0.00	0.40	0.00	0.00	0.00
fast	0		0.70	0.00	0.00	0.00	0.00
indexing	1		0.00	0.00	0.40	0.40	0.00
latent	1		0.00	0.00	0.40	0.00	0.40
learning	0		0.00	0.00	0.70	0.00	0.00
lsi	0		0.70	0.00	0.00	0.00	0.00
semantic	1		0.00	0.40	0.40	0.00	0.00
structures	0		0.00	0.00	0.00	0.40	0.40
tracks	0		0.70	0.00	0.00	0.00	0.00
tutorials	0		0.70	0.00	0.00	0.00	0.00

Table 3 shows cosine similarity results.

Table 3. Cosine similarity results.

Measure	Datum	Formula	d_1	d_2	d_3	d_4	d_5
Dot Product	$\mathbf{d_j} \bullet \mathbf{q}$	$\sum_{i=1}^{n} w_{i,j} w_{i,q}$	0.00	0.40	1.19	0.40	0.40
$\sum_{i=1}^{n} w_{i,q}^2$	3.00	$\sum_{i=1}^{n} w_{i,j}^2$	1.95	0.81	0.96	2.27	0.47
$\sqrt{\sum_{i=1}^{n} w_{i,q}^2}$	1.73	$\sqrt{\sum_{i=1}^{n} w_{i,j}^2}$	1.40	0.90	0.98	1.51	0.69
Magnitude Product	$\lVert\mathbf{d_j}\rVert \, \lVert\mathbf{q}\rVert$	$\sqrt{\sum_{i=1}^{n} w_{i,j}^2} \sqrt{\sum_{i=1}^{n} w_{i,q}^2}$	2.42	1.55	1.70	2.61	1.19
Cosine Similarity	$\frac{\mathbf{d_j} \bullet \mathbf{q}}{\lVert\mathbf{d_j}\rVert \, \lVert\mathbf{q}\rVert}$	$sim(d_j, q) = \frac{\sum_{i=1}^{n} w_{i,j} \, w_{i,q}}{\sqrt{\sum_{i=1}^{n} w_{i,j}^2} \sqrt{\sum_{i=1}^{n} w_{i,q}^2}}$	0.00	0.26	0.70	0.15	0.33

The documents rank in the following order of cosine similarities: $d_3 > d_5 > d_2 > d_4 > d_1$. These calculations can be replicated by hand or by properly using Excel built-in formulas SUMSQ, SQRT, and SUMPRODUCT. One may also combine these formulas to reduce the size of the table.

Additionally, the results can be double-checked with our Cosine Similarity Calculator (http://www.minerazzi.com/tools/cosine-similarity/cosine-similarity-calculator.php). With this tool, users may compute one cosine similarity at a time by submitting the term weights of a document and query vector.

User may compute cosine similarities in two different modes: centered and raw. If using this tool, you may want to use the default mode (raw). The other mode (centered) is used to convert a cosine similarity into a Pearson's correlation coefficient and to delve into the nature of the variables (Garcia, 2016b). These topics are out of the scope of this tutorial.

Conclusion

We have presented an introductory tutorial for those interested in learning about vector space models but that lack of a linear algebra background.

The calculations involved can be carried out with a spreadsheet, online calculator, or by hand. Thus, we believe that this tutorial might be suitable for students across disciplines.

Exercises

1. Rank the documents given in this tutorial, this time

    a. including all stopwords during indexing.
    b. using the Term Count Model for both document and query terms.
    c. using the TF-IDF Model for both document and query terms.

2. A database consists of 1 million documents, of which 200,000 contain the term *holiday* while 250,000 contain the term *season*. A document repeats *holiday* 7 times and *season* 5 times. It is known that *holiday* is repeated more than any other term in the document. Calculate the weight of both terms in this document using the following TF-IDF weighting schemes.

    a. $w_{i,j} = \left(\frac{f_{i,j}}{fmax_{i,j}}\right) IDF_i = \left(\frac{f_{i,j}}{fmax_{i,j}}\right) log\left(\frac{D}{d_i}\right)$
    b. $w_{i,j} = \left(\frac{f_{i,j}}{fmax_{i,j}}\right) IDFP_i = \left(\frac{f_{i,j}}{fmax_{i,j}}\right) log\left(\frac{D-d_i}{d_i}\right)$

    where $IDFP_i$ stands for *probabilistic inverse document frequency*.

References

Baeza-Yates, R. and Ribeiro-Neto, B. (1999). Modern Information Retrieval. Adisson Wesley. Book Review. Retrieved from http://www.amazon.com/gp/customer-reviews/R2HC8ULDSMXKZQ/ref=cm_cr_arp_d_rvw_ttl?ie=UTF8&ASIN=020139829X

Chisholm, E. and Kolda, T. G. (1999). New Term Weighting Formulas for the Vector Space Method in Information Retrieval. Oak Ridge National Laboratory. Retrieved from http://www.sandia.gov/~tgkolda/pubs/pubfiles/ornl-tm-13756.pdf

Garcia, E. (2016a). A Linear Algebra Approach to the Vector Space Model – A Fast Track Tutorial. Retrieved from http://www.minerazzi.com/tutorials/term-vector-linear-algebra.pdf

Garcia, E. (2016b). A Cosine Similarity Tutorial. Retrieved from http://www.minerazzi.com/tutorials/cosine-similarity-tutorial.pdf

Grossman, D. A., Frieder, O. (2004). Information Retrieval: Algorithms and Heuristics. Springer. Book Review. Retrieved from http://www.amazon.com/review/RACNGPXD2GNE7/ref=cm_cr_dp_title?ie=UTF8&ASIN=1402030045&channel=detail-glance&nodeID=283155&store=books

Rijsbergen, K. (2004). The Geometry of Information Retrieval. Cambridge University Press, UK. Book Review. Retrieved from http://www.amazon.com/review/R3FM04FS4ZDHGC/ref=cm_cr_dp_title?ie=UTF8&ASIN=0521838053&channel=detail-glance&nodeID=283155&store=books

Salton, G. & Buckley, C. (1987). Term Weighting Approaches in Automatic Text Retrieval.87-881. Cornell University. Retrieved from https://ecommons.cornell.edu/bitstream/handle/1813/6721/87-881.pdf?sequence=1&isAllowed=y
See also http://www.cs.odu.edu/~jbollen/IR04/readings/article1-29-03.pdf