

Term Vector Calculations

A Fast Track Tutorial

Dr. Edel Garcia
admin@miislita.com

First Published on November 3, 2005; Last Update: September 11, 2006.
Copyright © Dr. E. Garcia, 2006. All Rights Reserved.

Abstract

This fast track tutorial illustrates how term vector theory is used in Information Retrieval (IR). The tutorial covers term frequencies, inverse document frequencies, term weights, dot products, vector magnitudes and cosine similarities between documents and queries.

Keywords

term frequencies, inverse document frequencies, term weights, dot products, vector magnitudes, cosine similarities.

Problem

A query consisting of the two words “gift” and “card ” is submitted to a hypothetical collection of 100,000,000 documents. After removing all frequently used terms (stop words) these are the only unique terms, so a visual two-dimensional representation of the problem is possible.

Note: Almost all IR textbooks (e.g., Modern Information Retrieval, The Geometry of Information Retrieval, Information Retrieval – Algorithms and Heuristics, and others) illustrate term vectors in two and three dimensions (one per unique terms). This is done to help students visualize the problem. Evidently, with multiple dimensions a visual representation is not possible. In this case, a linear algebra approach is needed. This is described in

<http://www.miislita.com/information-retrieval-tutorial/term-vector-linear-algebra.pdf>

It is known that

- “gift” occurs in DOC1 2 times, in DOC2 1 time and in the collection in 300,000 documents.
- “card” occurs in DOC1 3 times, in DOC2 6 times and in the collection in 400,000 documents.

Assume term weights are defined using the probabilistic model

$$w = tf * IDF = tf * \log((N - n)/n)$$

where

tf = term frequency = number of times a term is repeated.

IDF = inverse document frequency = $\log((N - n)/n)$

n = number of documents containing a term.

Which one, DOC1 or DOC2, is a better match for the query?

Solution

Step 1. Compute term weights.

In DOC1

for “gift”: tf = 2 and IDF = $\log((100,000,000 - 300,000)/300,000) = 5.0431$

for “card”: tf = 3 and IDF = $\log((100,000,000 - 400,000)/400,000) = 7.1886$

Table 1 shows similar weight calculations for DOC2 and the Query.

Item	Term	tf	N	n	IDF	w
DOC1	gift	2	100,000,000	300,000	2.5216	5.0431
DOC1	card	3	100,000,000	400,000	2.3962	7.1886
DOC2	gift	1	100,000,000	300,000	2.5216	2.5216
DOC2	card	6	100,000,000	400,000	2.3962	14.3772
Query	gift	1	100,000,000	300,000	2.5216	2.5216
Query	card	1	100,000,000	400,000	2.3962	2.3962

Table 1. Term weight calculations for DOC1, DOC2 and Query.

If DOC1, DOC2 and Query are represented as points in a gift-card term space, then these weights are coordinate values defining the points. This is illustrated in Figure 1.

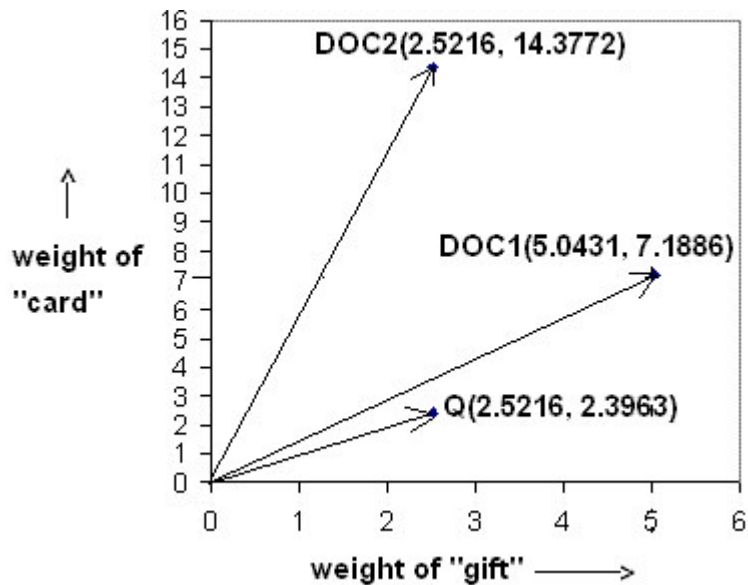


Figure 1. Term space for DOC1, DOC2 and Query.

Step 2. Compute DOT products between points.

For two dimensions, x and y, the DOT Product is given by $x_1 \cdot x_2 + y_1 \cdot y_2$

$$\text{DOC1} \bullet \text{Query} = 5.0431 \cdot 2.5216 + 7.1886 \cdot 2.3962 = 29.94200$$

$$\text{DOC2} \bullet \text{Query} = 2.5216 \cdot 2.5216 + 14.3772 \cdot 2.3962 = 40.8091$$

By comparing Dot Products only, we may think that DOC2 is a better match for Query than DOC1.

Step 3. Compute vector magnitudes (Euclidean distance of each point from the origin of the term space)

For two dimensions, x and y, the Euclidean distance is given by $(x_1^2 + y_1^2)^{1/2}$

$$|\text{DOC1}| = (5.0431 \cdot 5.0431 + 7.1886 \cdot 7.1886)^{1/2} = (25.4329 + 51.6760)^{1/2} = 8.78117$$

$$|\text{DOC2}| = (2.5216 \cdot 2.5216 + 14.3772 \cdot 14.3772)^{1/2} = (6.3585 + 206.7039)^{1/2} = 14.5967$$

$$|\text{Query}| = (2.5216 \cdot 2.5216 + 2.3962 \cdot 2.3962)^{1/2} = (6.3585 + 5.7418)^{1/2} = 3.47854$$

Step 4. Compute cosine similarities between Query and each document by dividing the DOT products by the product of the corresponding magnitudes. This corrects for document lengths.

For DOC1 and Query

$$\text{Cosine angle} = \text{DOC1} \bullet \text{Query} / (|\text{DOC1}| * |\text{Query}|) = 29.94200 / (8.78117 * 3.47854) = 0.9802$$
$$\text{Cosine angle} = \text{DOC2} \bullet \text{Query} / (|\text{DOC2}| * |\text{Query}|) = 40.8091 / (14.5967 * 3.47854) = 0.80372$$

Step 5. Draw conclusions.

DOC1 is a better match for Query than DOC2. Haven't normalized the DOT products we have assumed the opposite (as given in Step 2). Note also from Figure 1 that DOC2 repeats "gift" 6 times, which causes its vector to be displaced away from the Query vector. In this case, term repetition has a detrimental effect on the document.

Questions

1. What would the results above look like if we have used term frequency to define term weights (without multiplying by IDF)?
2. Assuming that cosine values below 0.90 are considered a "bad" match, calculate the similarity between DOC1 and DOC2. Are they a good match?
3. In this fast track tutorial we treated a query as another document. We also used cosine normalization. What are the main drawbacks of these approaches?
4. Repeat this fast track tutorial using the original data, but this time assuming that $N = 700,000$. What happens with the weight of "card" and why? How does this could affect retrieval and why?
5. A term weight model defines term weights for queries as

$$w = 0.5 + 0.5 * \text{tf} / \text{tf}_{\max}$$

where tf is the usual term frequency and tf_{\max} is the maximum term frequency of a term in a query. Term weights for documents are computed using

$$w = \text{tf} * \log((N - n) / n).$$

Another model defines term weights for both documents and queries as

$$w = (0.5 + 0.5 * \text{tf} / \text{tf}_{\max}) * \log((N - n) / n)$$

Redo this fast track tutorial using a query where “gift” is repeated 2 times and “card” is repeated 1 time. Compare both models.

References

1. <http://www.miislita.com>
2. <http://www.miislita.com/term-vector/term-vector-1.html>
3. <http://www.miislita.com/term-vector/term-vector-4.html>
4. <http://www.miislita.com/information-retrieval-tutorial/cosine-similarity-tutorial.html>

Copyright © Dr. E. Garcia, 2006. All Rights Reserved