The Binary and Term Count Models

Abstract – This is Part 2 of an introductory tutorial series on Term Vector Theory as used in Information Retrieval and Data Mining. The Binary (BNRY) and Term Count (FREQ) models are discussed.

Keywords: binary model, term count model, vectors term vector theory, term weights, local weights
Published:10-27-2006; Updated: 03-18-2016

Note: This article is part of a legacy series that the author published circa 2006 at http://www.miislita.com, now a search engine site. It is now republished in pdf format here at http://www.minerazzi.com, with its content edited and updated. The original articles can be found referenced in online research publications on IR and elsewhere.

Introduction

In Part 1 of this series on vector space models (Garcia, 2016a) it was mentioned that in Information Retrieval (IR) local and global information are used to score term weights

$$w_{i,j} = L_{i,j} G_i \tag{1}$$

where $L_{i,j}$ accounts for the presence of a term in a document and G_i across the collection.

The difference between the several models depends on how $L_{i,j}$ and G_i are defined (Baeza-Yates & Ribeiro-Neto, 1999; Grossman & Frieder, 2004; Rijsbergen, 2004). In the next sections, we discuss two of these models.

The Binary Model (BNRY)

The simplest model one can think of is a binary model (BNRY) where local weights are considered independent of term frequencies and where global weights are ignored.

$$w_{i,j} = L_{i,j} \begin{cases} 1 \; if \; f_{i,j} > 0 \\ 0 \; if \; f_{i,j} = 0 \end{cases} \tag{2}$$

In (1), $w_{i,j} = L_{i,j} = 1$ if the term is in the document; otherwise, $w_{i,j} = L_{i,j} = 0$.

BNRY is recommended for pre-weighting or quickly scanning a small index and for scoring small collections of short titles, abstracts, and documents. As documents of different lengths are equally weighted, it is a low precision model (Salton & Yang, 1973) in the sense that it cannot discriminate between relevant and non-relevant results.

Because of that, a retrieval system using BNRY will frequently find vocabulary-rich documents simply because they happen to mention query terms. Thus, the model can be easily gamed by automatically generating documents or pseudo-documents with frequently queried terms. This is one of the many forms of *spamdexing* used across the Web (AIRWeb, 2007).

We can improve the model by making local weights a linear function of term frequencies. This leads us to the Term Count Model.

The Term Count Model (FREQ)

The main assumption behind this model is that a document repeating a term several times is likely to be relevant to said term. This idea was first proposed by Luhn and investigated by Salton and Yang (Luhn, 1953; 1957; Salton & Yang, 1973; Salton, 1983).

So by making local weights a linear function of term frequencies

$$w_{i,j} = L_{i,j} = f_{i,j} \tag{3}$$

one should be able to improve retrieval precision by finding relevant results at the top of the search results. This is what the Term Count Model, also known as FREQ (Chisholm & Kolda, 1999), tries to accomplish.

Unfortunately, assuming a linear relationship function between local weights and term frequencies ($L_{i,j} = f_{i,j}$) is not a best matching approach, but can be exploited by simply repeating a term. This is another form of *spamdexing* known as *keyword stuffing* (AIRWeb 2007).

One way of avoiding term repetition abuses consists in assigning a variable weight to different instances of a given term. This is what *Best Matching (BM) Algorithms* try to accomplish, the most famous of these is *BM25* and its many variants (Wikipedia, 2016). These algorithms are discussed in another tutorial series.

The Vector Space

Regardless of the weighting scheme used, documents and queries can be represented as objects (points or vectors) in an *n*-dimensional space where each term is a dimension. So a document d_j with *n* number of terms can be represented as a point or vector with coordinates $d_j(w_{1,j}, w_{2,j} \ldots w_{n,j})$. As a query is like another document, its coordinates in said space are $q(w_{1,q}, w_{2,q} \ldots w_{n,q})$.

A vector is a quantity with direction and magnitude. The direction of a vector relative to other vectors is obtained by multiplying their coordinates. The result is a quantity called the dot product, $\mathbf{d_j \bullet q}$. Thus, the dot product between a document and a query is obtained by multiplying the coordinates $d_j(w_{1,j}, w_{2,j} \ldots w_{n,j})$ and $q(w_{1,q}, w_{2,q} \ldots w_{n,q})$ and adding together the products,

$$\mathbf{d_j \bullet q} = w_{1,j} * w_{1,q} + w_{2,j} * w_{2,q} \ldots w_{n,j} * w_{n,q} = \sum_{i=1}^{n} w_{i,j}\, w_{i,q} \quad (4)$$

Figure 1. shows the dot product between a document mentioning [auto] and [insurance] three times each and [car] once. The query consists of the term [insurance]. The document vector is at ***d(3, 1, 3)*** and the query vector at ***q(0, 0, 1)***.

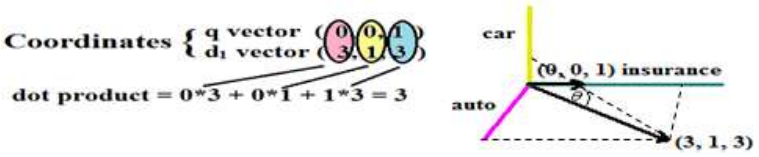

Figure 1. Document and query vectors computed with the Term Count Model.

The magnitude of a vector is simply its Euclidean length, L, also known as the L_2-norm, and computed by squaring individual coordinates, adding them together, and taking square roots. Thus for a document and query vectors, $\mathbf{d_j}$ and $\mathbf{q}$, their absolute magnitudes are

$$L_{d_j} = \sqrt{{w_{1,j}}^2 + {w_{2,j}}^2} = \sqrt{\sum_{i=1}^{n} w_{i,j}^2} = ||\mathbf{d_j}|| \quad (5)$$

$$L_q = \sqrt{{w_{1,q}}^2 + {w_{2,q}}^2} = \sqrt{\sum_{i=1}^{n} w_{i,q}^2} = ||\mathbf{q}|| \quad (6)$$

Dividing (4) with (5) and (6), we obtain the cosine of the angle, *cos(θ)*, between document and query vectors. As the two vectors approach each other, the angle between them, *θ*, decreases and the cosine of the angle, *cos(θ)*, increases. If the two vectors are superimposed, *cos(θ)* = 1, and the two vectors are fully similar to one another.

When computed to compare d_j and q, *cos(θ)* is taken for a resemblance measure and referred to as the *cosine similarity* between d_j and q, $sim(d_j, q)$,

$$sim(d_j, q) = \frac{\mathbf{d_j} \bullet \mathbf{q}}{\|\mathbf{d_j}\| \, \|\mathbf{q}\|} = \frac{\sum_{i=1}^{n} w_{i,j} \, w_{i,q}}{\sqrt{\sum_{i=1}^{n} w_{i,j}^2} \sqrt{\sum_{i=1}^{n} w_{i,q}^2}} \tag{7}$$

As cosines are not additive (i.e., a sum of cosines is not a cosine), they cannot be arithmetically averaged. Therefore, computing arithmetic averages from cosine similarities is a misleading flawed practice.

As shown later in this tutorial, (7) can also be obtained by computing dot products from unit vectors; i.e., by normalizing vector elements with the vector length before computing their dot products. This is the so-called *cosine normalization*. To indicate its usage, (1) is rewritten as

$$w_{i,j} = L_{i,j} G_i N_j \tag{8}$$

where

$$N_j = \frac{1}{\sqrt{\sum_{i=1}^{n} \left(L_{i,j} G_i\right)^2_{i,j}}} \tag{9}$$

Other forms of weight normalizations have been proposed, though (Singhal, Buckley, & Mitra, 1996a; Singhal, Salton, Mitra & Buckley, 1996a; Singhal, Salton & Buckley, 1996c; Chisholm & Kolda, 1999). Still, the Binary and Term Count models still tend to favor long documents simply because they are longer (Lee, Chuang, & Seamons, 1997).

Cosine Similarity Calculations

Table 1 shows the result of scoring a collection of three documents mentioning the terms [auto], [car], or [insurance] with the query [insurance] using the Term Count Model (FREQ).

Table 1. Term Count Model Results for 3 documents.

Index terms	q		d_1	d_2	d_3
auto	0		3	1	2
car	0		1	2	3
insurance	1		3	4	0
		$\sum_{i=1}^{n} w_{i,j} w_{i,q}$	3	4	0
$\sum_{i=1}^{n} w_{i,q}^2$	1	$\sum_{i=1}^{n} w_{i,j}^2$	9	21	13
$\sqrt{\sum_{i=1}^{n} w_{i,q}^2}$	1	$\sqrt{\sum_{i=1}^{n} w_{i,j}^2}$	4.36	4.58	3.61
		$\sqrt{\sum_{i=1}^{n} w_{i,j}^2}\sqrt{\sum_{i=1}^{n} w_{i,q}^2}$	4.36	4.58	3.61
		$sim(d_j, q) = \frac{\sum_{i=1}^{n} w_{i,j}\, w_{i,q}}{\sqrt{\sum_{i=1}^{n} w_{i,j}^2}\sqrt{\sum_{i=1}^{n} w_{i,q}^2}}$	0.69	0.87	0

As expected,

$$sim(d_1, q) = \frac{\sum_{i=1}^{n} w_{i,j}\, w_{i,q}}{\sqrt{\sum_{i=1}^{n} w_{i,j}^2}\sqrt{\sum_{i=1}^{n} w_{i,q}^2}} = \frac{3}{4.36 * 1} = 0.69$$

$$sim(d_2, q) = \frac{\sum_{i=1}^{n} w_{i,j}\, w_{i,q}}{\sqrt{\sum_{i=1}^{n} w_{i,j}^2}\sqrt{\sum_{i=1}^{n} w_{i,q}^2}} = \frac{4}{4.58 * 1} = 0.87$$

$$sim(d_3, q) = \frac{\sum_{i=1}^{n} w_{i,j}\, w_{i,q}}{\sqrt{\sum_{i=1}^{n} w_{i,j}^2}\sqrt{\sum_{i=1}^{n} w_{i,q}^2}} = \frac{0}{3.61 * 1} = 0$$

Therefore, the documents are ranked in the following order: $d_2 > d_1 > d_3$.

A Linear Algebra Approach

In a recent tutorial, we discussed a linear algebra approach that greatly simplifies vector space calculations (Garcia, 2016b).

Essentially, the matrix $\mathbf{q}^T\mathbf{A}$ is computed where $\mathbf{q}^T$ is the transpose of $\mathbf{q}$ and $\mathbf{A}$ is a matrix of unit vectors $\hat{\mathbf{d}}_j$. A unit vector, denoted with a hat (^), is obtained by dividing a vector elements by its magnitude (L_2-norm or Euclidean length). Thus, $\mathbf{q}^T\mathbf{A}$ is a matrix filled with cosine similarities equal to dot products. The $\mathbf{q}$, $\mathbf{A}$, and $\mathbf{q}^T\mathbf{A}$ matrices obtained from the data shown in Table 1 are

Index terms		$\hat{\mathbf{q}}$		$\hat{\mathbf{d}}_1$	$\hat{\mathbf{d}}_2$	$\hat{\mathbf{d}}_3$
auto		0		0.69	0.22	0.55
car	$\mathbf{q}$ =	0	$\mathbf{A}$ =	0.23	0.44	0.83
insurance		1		0.69	0.87	0.00

	d_1	d_2	d_3
$\mathbf{q}^T\mathbf{A}$ = [	0.69	0.87	0]

Notice that the documents are ranked as before. To sum up, the so-called *cosine normalization* is another way to say that we are computing the dot product between unit vectors. When unit vectors are multiplied, by definition their dot product equals the cosine of the angle between them.

Another way of achieving the same results consists in augmenting $\mathbf{A}$ with the unit vector of the query and then computing the $\mathbf{A}^T\mathbf{A}$ similarity matrix. The query vector can be placed as the first column vector, like this

Index terms		$\hat{\mathbf{q}}$	$\hat{\mathbf{d}}_1$	$\hat{\mathbf{d}}_2$	$\hat{\mathbf{d}}_3$
auto		0	0.69	0.22	0.55
car	$\mathbf{A}$ =	0	0.23	0.44	0.83
insurance		1	0.69	0.87	0.00

	q	d_1	d_2	d_3	
	1.00	0.69	0.87	0.00	q
$\mathbf{A}^T\mathbf{A}$ =	0.69	1.00	0.85	0.57	d_1
	0.87	0.85	1.00	0.48	d_2
	0.00	0.57	0.48	1.00	d_3

Augmenting **A** in this way can be justified as in the vector space the query vector behaves like a document vector. For the purpose of comparing vector similarities, making the query vector the first, last, or a given column vector of **A** does not really matter.

So what do we gain from computing $\mathbf{A}^T\mathbf{A}$? First, document ranking results are readily computed, in this case from the first row or column of $\mathbf{A}^T\mathbf{A}$. Second, a straightforward comparison of document vector similarities is possible. In this example,

$sim(d_1, d_2) = 0.85$
$sim(d_1, d_3) = 0.57$
$sim(d_2, d_3) = 0.48$

That is, d_1 and d_2 are the most similar documents.

Conclusion

The advantages and limitations of the Binary (BNRY) and Term Count (FREQ) models have been covered. Both models are based on computing local weights, ignoring global information.

BNRY ignores term frequencies, but FREQ does not. Both models ignore global information, tend to favor long document, and are vulnerable to spamdexing. As both are based on matching terms, documents not mentioning query terms, but their synonyms, are not retrieved even if these are relevant to the query. This is a common drawback found in most vector space models.

Exercises

1. Rework this tutorial exercise, this time by defining $L_{i,j}$ in (3) as follows where for a given document $maxf_{i,j}$ is its maximum term frequency and $avef_{i,j}$ its average term frequency. See Chisholm & Kolda (1999).

   - $L_{i,j} \begin{cases} \frac{f_{i,j}}{maxf_{i,j}} > 0 \\ 0 \; if \; f_{i,j} = 0 \end{cases}$
   - $L_{i,j} \begin{cases} \frac{f_{i,j}}{avef_{i,j}} > 0 \\ 0 \; if \; f_{i,j} = 0 \end{cases}$

References

AIRWeb (2007). Adversarial Information Retrieval on the Web. Retrieved from http://airweb.cse.lehigh.edu/2007/cfp.html

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). Term Vector Theory and Keyword Weights. Retrieved from http://www.minerazzi.com/tutorials/term-vector-1.pdf

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

Lee, D. L., Chuang, H., and Seamons, K. (1997). Document Ranking and the Vector-Space Model. IEEE March/April, pp 67-75. Retrieved from http://www.cs.ust.hk/faculty/dlee/Papers/ir/ieee-sw-rank.pdf

Luhn, H. P. (1953). A New Method of Recording and Searching Information. Retrieved from http://jonathanstray.com/papers/Luhn-SearchEngine-1953.pdf

Luhn, H. P. (1957). A Statistical Approach to Mechanized Encoding and Searching of Literary Information. IBM Journal. Retrieved from http://web.stanford.edu/class/linguist289/luhn57.pdf

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. (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill.

Salton, G., Wong, A., Yang, C. S. (1975). A Vector Space Model for Automatic Indexing. Communications of the ACM 18 (11): 613. Retrieved from http://elib.ict.nsc.ru/jspui/bitstream/ICT/1230/1/soltan_10.1.1.107.7453.pdf
see also http://www.bibsonomy.org/bibtex/10a4c67f15a4869634d8e5e39ba3e7113

Salton, G. and Yang, C. S. (1973). On the Specification of Term Values in Automatic Indexing. TR 73-173, Cornell University. Retrieved from https://ecommons.cornell.edu/bitstream/handle/1813/6016/73-173.pdf?sequence=1&isAllowed=y

Singhal, S., Buckley, C., and Mitra, M (1996a). ACM SIGIR'96, 21-29, 1996. Pivoted Document Length Normalization. Retrieved from http://singhal.info/pivoted-dln.pdf

Singhal, S., Salton, G., Mitra, M., and Buckley, C. (1996b). Document Length Normalization. Information Processing and Management, 32:5, 619-633, 1996. Retrieved from https://ecommons.cornell.edu/bitstream/handle/1813/7186/95-1529.pdf?sequence=1

Singhal, A., Salton, G., and Buckley, C. (1996c). Length Normalization in Degraded Text Collections. Fifth Annual Symposium on Document Analysis and Information Retrieval, 149-162, 1996. Retrieved from http://singhal.info/ocr-norm.pdf

Wikipedia (2016). Okapi BM25. Retrieved from https://en.wikipedia.org/wiki/Okapi_BM25