INFO 630 / CS 674 Lecture Notes

Pivoted Document Length Normalization

Lecturer: Lillian Lee
Lecture 3: September 4, 2007
Scribes: Vladimir Barash, Stephen Purpura, Shaomei Wu

Introduction and Motivation

Today's lecture notes cover pivoted document length normalization by Singhal, Buckley, and Mitra from SIGIR '96. Before we dive into the details, we will review our classic VSM (vector space model) ad hoc information retrieval derivation[1].

Our information retrieval goal is to rank documents (which are elements of a corpus) by the notion of relevance to the query, q. The query, q, expresses the user's information needs. The vector space model represents documents (*d*) in a corpus as vectors, each entry of which corresponds to a term. Each term is an element of the corpus *vocabulary*. The vocabulary might be the set of all words, phrases, or units of observation occurring in the document, but sometimes the set of terms is more restricted (see Porter Stemming for an example). The document vector's elements are term weights, d[1],...,d[m], with each element corresponding to a weight for the document's use of vocabulary terms $v_1,...,v_m$.

$$(Eq\,1):\ \vec{d}=\begin{bmatrix} d[1] \\ ... \\ d[m] \end{bmatrix}=(d[1]...d[m])^T$$

The document vector's term weights are, by consensus, formed from three components:

$$(Eq\,2):\ d[j]=\frac{[tf_d(j)idf(j)]}{norm(d)}$$

In the above equation,

- $tf_d(j)$ is some function based on the term frequency within the document d
- idf(j) is inversely related to the number of documents in C that contain v_j

The term-weighting component is intended to measure whether the term is a good characterizer for the document, and it is used within our match (or scoring) function:

$$(Eq\,3):\ \sum_{j=1}^{m} q[j]d[j]$$

Unfortunately, the term-weighting scheme we have developed can unfairly advantage long documents in two ways:

(1) term frequency 'tf' counts are bigger in larger documents because there is a larger pool of

word positions to choose from.
(2) there are more non-zero term frequencies ('tf') because the probability of any word in the vocabulary appearing in a long document increases relative to short documents.

We wish to avoid a match function which has a bias towards producing higher relevance rankings for long documents over shorter documents, if the shorter documents are actually more relevant. Therefore, normalization is used to compensate within the match function for this bias.

In the last lecture, we reviewed two normalization methods, L_1-normalization and L_2-normalization, for correcting the match function for the bias. L_2-normalization was shown to be more useful than L_1-normalization. In this lecture, we examine "pivoted document length normalization" from [SBM '96]. The paper is an interesting example of empirical research conducted by graduate students at Cornell, because researchers confronted the question of whether L_2-normalization was the best engineering solution for achieving the user's information retrieval goal. Using empirical research methods, the researchers investigate and conclude that "better retrieval effectiveness results when a normalization strategy retrieves documents with chances similar to their probability of relevance." [SBM '96]

L_2-Normalization Review

We (and [SBM '96]) notice that the choice of normalization function is partially based on theory, but mostly attuned to achieving high performance in our information retrieval goal of highly ranking the most relevant documents to match the query. In this sense, prior to [SBM '96] the assumption was that L_2-normalization was performing well. [SBM '96] investigates how well L_2-normalization performed in practice on the TREC corpora, and it proposes a new normalization function, "pivoted document normalization", which the authors demonstrate is better at achieving the enumerated user information need as specified for a subset of the TREC corpora.

Recall that our normalization function, norm(d), can be considered the length penalty which addresses the two problems of bias caused by long documents. In L_2-Normalization, norm(d) is set as follows:

$$(Eq\,4):\quad norm(d)=\sqrt{\sum_{j}(raw\,tf\,idf[j])^2}$$

Our norm(d) is applied to terms in the match function but it is (A) term independent and (B) document dependent. Our norm(d) function corresponds to cosine scoring and cosine scoring seems reasonable, and not obviously refutable. At the time of [SBM '96], L_2 was a common normalization function in the information retrieval literature.

Empirically Validating the Performance of the L_2-Normalization Function

The first task in [SBM '96] is to empirically check whether the performance of the norm function (within the context of the term-weighting function and match function) is 'well fit'. More concretely:

> ***How does the length distribution of (truly) relevant documents compare to the length distribution of retrieved documents, with respect to L_2-normalization?***

The plots below in Figure 1 from [SBM '96] seek to highlight the comparison. First, 741,856 documents from the TREC corpora were ranked in order of file byte size and then divided into 742 bins of 1,000 documents (the final bin with the largest file sizes had 856 documents). For the plots in Figure 1, the median document length of each bin was used to generate a point for the bin. Next, 9805 'relevant' query-document pairs (q,d) were generated by finding where the document d was judged relevant to a query q for 50 TREC queries matched across the 741,856 documents.

Since the objective of normalization is to compensate for ranking bias caused by document length,

Figure 1 helps us clarify the relationship between relevance and document length in real-world corpora. Graph (a) shows that the probability increases with file size that a relevant document will be in a bin with a larger file size. Long documents do have higher relevance compared with short documents, which can be explained by the fact that long documents usually cover more topics and have broader content. Graph (b) shows that the probability that a relevant document is retrieved using L_2-normalization follows a similar pattern to Graph (a), but the probability of retrieval for larger documents does not increase as rapidly as the probability of relevance expressed in graph (a). Graph (c) shows the implications of graphs (a) and (b). The penalty imposed by L_2-normalization on relevant documents with document length larger than a pivot point (p) is actually greater than desired. Documents shorter than pivot point (p) have a greater probability of retrieval than is warranted for their probability of relevance.

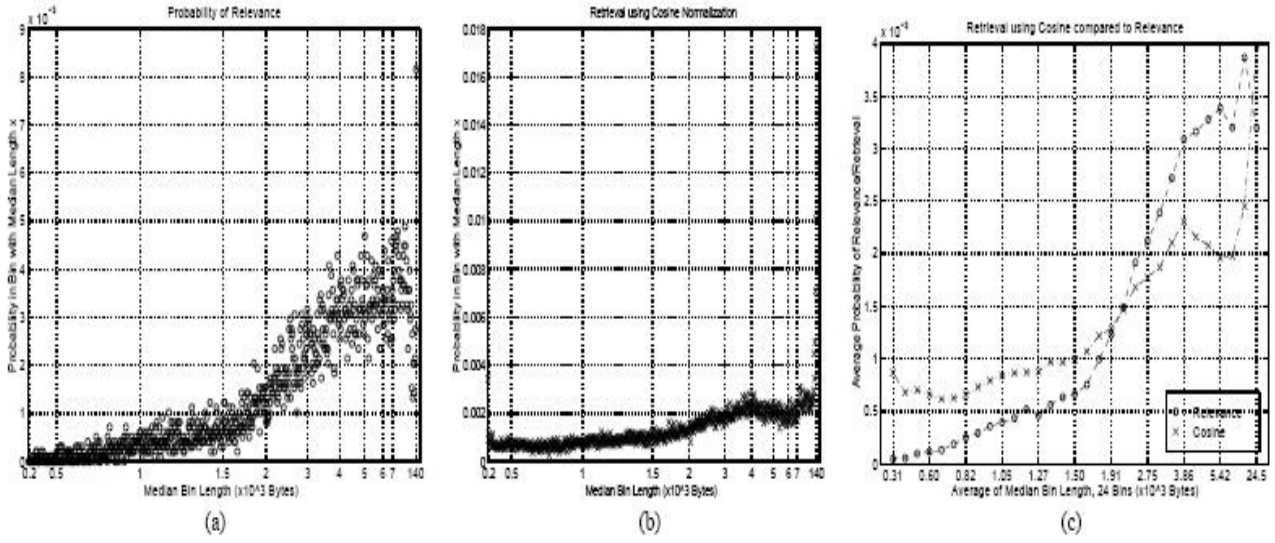


Figure 1: Probability that a relevant/retrieved document is from a bin, plotted against the median bin length. The analysis for the relevant documents is shown in (a), (b) shows the analysis for documents retrieved using cosine normalization, and (c) compares the smooth plots for (a) and (b).

It is interesting to note that the x-axis in the plots of Figure 1 are measured in bytes because they are a more neutral measure than words. Using bytes also saves the time required to open the documents and count the words or lines. The y-axis for graphs (a) and (b) show the probability that a given document is in a bin with median length x, given that d is an element of (q,d). Conceptually similar, the y-axis for graph (c) is the Average Probability of Relevance/Retrieval.

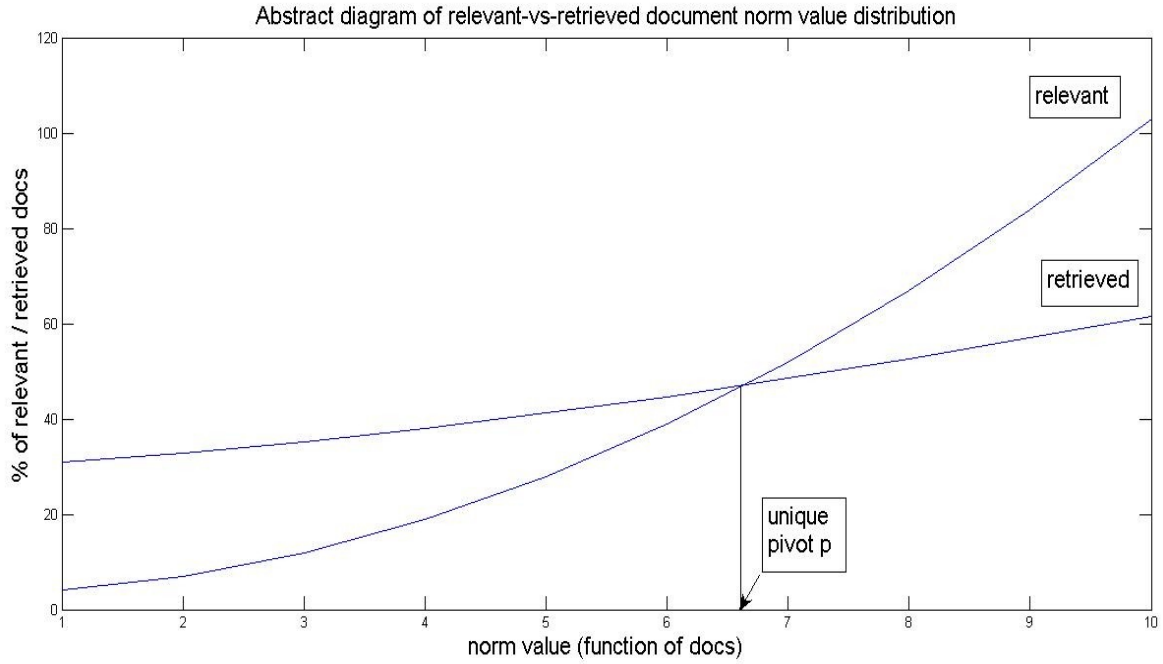
It is also worth mentioning that the researchers showed a good approach to shrink the x-axis and increase the smoothness of plot. Had they just plotted 9805 relevant documents out of 740,000, the graph would have been too sparse to form interesting curves. It is very likely that we would find very few or no documents for a lot of values of length, and such a plot wouldn't be as meaningful. So, [SBM '96] took the idea of "histogram" and grouped documents into length groups to increase the meaningfulness of each point on the plot. Each of these length groups, or "metabins", represents the average of every 24 consecutive bins.

Adopting the metabin approach is very important from a methodological perspective. On the one hand, compressing bins into metabins creates a smoother plot; on the other hand, it provides the researchers with data for both coarse-grained (metabinned) and fine-grained partitioning (binned) of document lengths. A fine-grained partitioning by itself would probably be too noisy to show any significant patterns, while a coarse-grained partitioning by itself would show too little noise, and raise suspicions about SBM's data analysis process in other researchers.

Simplification of the Empirical Observations

From Figure 1, we can see that the curve which approximates the probability of relevant/retrieved documents (using L_2-normalization) crosses the curve representing the probability of relevant documents. We call the crossing point of the two curves the pivot point. In addition, we observe the trends that retrieval using L_2-normalization over-prefers short documents and under-prefers long documents. But at some pivot point, p, it does match the distribution of retrieved documents.

For simplicity, we can model the behavior observed in [SBM '96]'s graph (c) of Figure 1 by reducing it to a smoother form as follows:



Improving Performance over the L_2-Normalization Function -- Pivoted Document Normalization

The empirical observations we discussed earlier imply that we could improve our match function's performance in achieving the user's information need if we had a "norm'(d)" (as a function of norm(d)) such that:

- if norm(d) == p, norm'(d) = norm(d)
- if norm(d) > p, norm'(d) < norm(d)
- if norm(d) < p, norm'(d) > norm(d)
- for all values of d, norm(d) is reasonably close to norm(d)

The last criterion is suggested by the fact that, even though our match function produces a relevant document length distribution that is different than the actual relevant document length distribution in TREC, the values of the two distributions are in the same order of magnitude (average p of relevance / retrieval $\sim 1x10^{-2}$)

However, in practice, there are challenges to developing norm'(d):

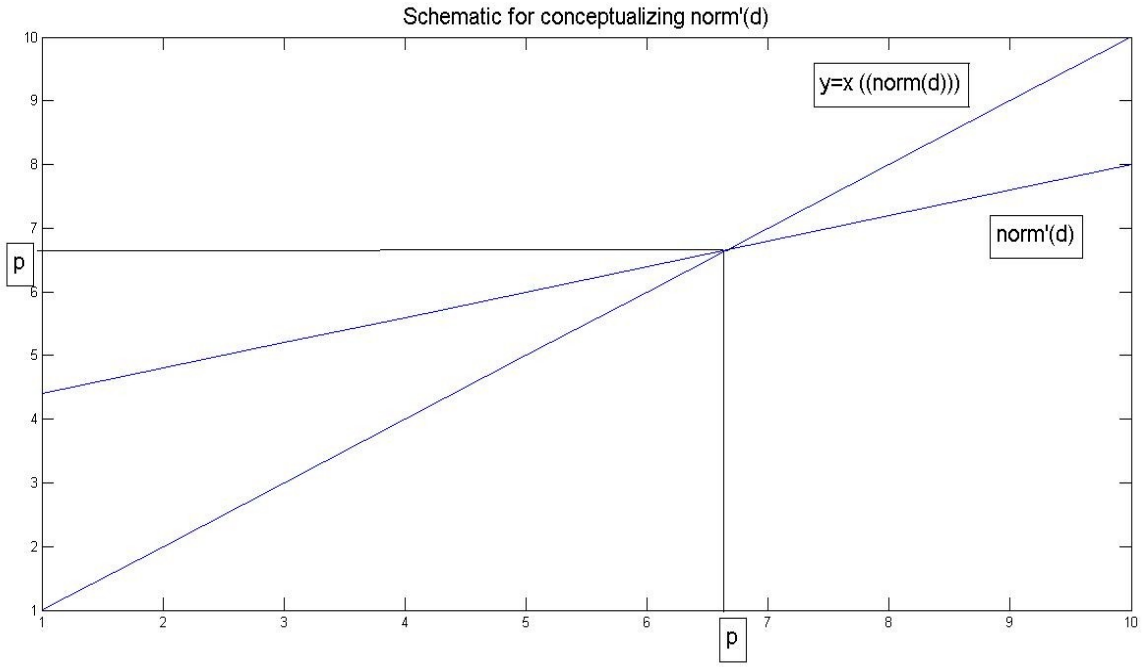
- Our specification for norm'(d) is not constrained enough.
- Little information is available to assist us. Our information retrieval function will not know the crossing point p or the relevance distribution.

To solve these problems, we want a "simple" norm'(d) (for instance, a linear function of d) with as few parameters as possible.

Considering some alternatives for basic functions of norm'(d):

1. We could choose norm'(d) = Constant, but this doesn't make sense because it doesn't compensate for the unfairness of document length.
2. *We could try a linear function* of norm(d):

$$(Eq\,5):\ \ norm'(d) = m'\,norm(d) + b'$$



The linear function looks like it might work to approximate our desiderata, but we would like to eliminate as many of the unknown parameters (m' and b') as possible.

Parameter Reduction

1. Apply known information:
    - if there is a value dp such that norm(dp) == p, we want norm'(dp) = p

$$(Eq\,6):\ \ m'\,norm(dp) + b' = p$$
$$(Eq\,7):\ \ m'\,p + b' = p$$
$$(Eq\,8):\ \ b' = p(1 - m')$$
$$(Eq\,9):\ \ norm'(d) = m'\,norm(d) + p(1 - m')$$

2. Use a test value of p:

$$(Eq\,10):\quad \overline{norm} = \frac{\sum_{d \in C} norm(d)}{|C|}$$

Why should p be set as norm-bar? [SBM '96] shows us that concrete value of p actually doesn't matter as much as we might think, because setting p to norm-bar is no more than a re-parametrization, and does not alter the ranking produced. We present the proof of this claim below, reproduced from SBM's paper:

Proof:

1. group unknowns:

$$(Eq\,11):\quad norm'(d) = \left[\!\!\left[\frac{m' norm(d) + p(1-m')}{p(1-m')} \right]\!\!\right] p(1-m')$$

$$\overset{rank}{=} \frac{m'}{p(1-m')} norm(d) + 1$$

The rank equality holds because p(1-m') is a document- and term-independent quantity.

2. pick some m'' such that p(1-m') == norm-bar(1-m''). Then, (11) is equal to:

$$(Eq\,12):\quad \frac{m''}{\overline{norm}(1-m'')} norm(d) + 1$$

because m'' / (1 - m'') is a bijection onto (0, inf).

3. get as interpolation:

$$(Eq\,13):\quad norm'(d) \overset{rank}{=} norm''(d)$$

$$= \frac{m'' norm(d)}{\overline{norm}} + (1-m'') * 1$$

Note that if norm(d) == norm-bar, norm''(d) = 1. Now we see why norm is a good choice for p. Rank equivalence shows that setting p to norm-bar is no more than a re-parametrization, QED

Conclusion

In vector-space-model information retrieval, the choice of normalization method matters. Choosing L1-normalization results in a bias towards documents containing few different types of terms (few non-zero term frequencies). L_2-normalization is more justifiable, correcting the match function for the bias of long documents. But "pivoted document length normalization" from [SBM '96] provides us with an empirical method to adjust for the bias of long documents without over-compensating. This method provides better retrieval effectiveness by using a normalization strategy which retrieves documents with chances similar to their probability of relevance.

References

See Pivoted Document Length Normalization. Amit Singhal, Chris Buckley, Mandar Mitra. *ACM SIGIR'96*, 21-29, 1996. Referenced as [SBM '96] in this document.

Footnotes

[1] We are using a non-standard notation because of the following irritating problem: if one used the standard notation, d_j might be either the j^{th} coordinate of the vector d or the j^{th} document of the set of documents, D. So, to distinguish between the two, d[j] (scalar) is the j^{th} coordinate of the vector **d** and d_j is the j^{th} document of the set of documents, D.

Lecture Vocabulary Terms

The following terms are useful for understanding the material:

- d: a document, an element of the set of D documents
- C: a corpus, a synonym for the set of D documents
- Vocabulary: the set of m terms used to represent documents in C
- v: a term in the Vocabulary; an element of V
- j: an index variable within a vector (esp. for vectors in R^m)
- term: an element of the set of Vocabulary; typically a word, phrase, or other unit of observation about a text
- q: a query expressing a user's information needs
- tf: term frequency (term occurrence in document)
- idf: inverse document frequency of a term
- norm(): a normalization function

In our notation, d and q are represented as vectors with index variable j.

INFO 630/CS 674 Lecture Notes: Finger Exercise

Scribes: Vladimir Barash, Stephen Purpura, Shaomei Wu

Here is a quick finger exercise to let you practice the L_1, L_2 and pivoted document length normalization schemes we presented in class.

Given a corpus with *m* terms, assuming each $tf_d(j), idf(j)$ is non-negative, here *d* is a document from the corpus.

The L_1-normalization function is:

$$norm_1(d) = \sum_{j=1}^{m} tf_d(j) \times idf(j)$$

The L_2-normalization function is:

$$norm_2(d) = \sqrt{\sum_{j=1}^{m} [tf_d(j) \times idf(j)]^2}$$

The pivoted document length normalization for L_2 normalization from [SBM '96] is:

$$norm'(d) = m' \times norm_2(d) + b' = m' \times norm_2(d) + p(1 - m')$$

In the following three part problem, we pick pivot point *p* to be the average value of $norm_2(d)$ over all the documents in our given corpus, and as recommended by [SBM '96][1], we let $m' = 0.2$.

Problem Part 1:

Given a corpus containing 4 documents, with the term frequencies over the corpus vocabulary as below:

Table 1: Corpus term-frequency table

	cat	dog	household	love	useful
d_1	2	2	0	1	0
d_2	2	2	0	0	0
d_3	0	4	4	1	3
d_4	8	6	2	6	4

[1] Setting m=0.2 is recommended in [SBM '96] for pivoted unique normalization. See [SBM '96] for a more complete discussion.

Given the query "love cat" using a binary query vector (i.e. $q = [1,0,0,1,0]^T$ over the vocabulary), calculate $norm_1$, $norm_2$ and $norm'$ for each document.

Part 1 Solution:

The "*tf*" term representing the term-frequency in each document is already given in Table 1. For the purposes of this exercise, we will use the following '*idf*' function (note this is different from the *idf* function that we have used in other places in the notes):

$$idf_d(j) = \ln\left(\frac{N}{n_j}\right),$$

where N is the size of the corpus (the total number of documents in the corpus) and n_j is the number of documents that contain term j.

Table 2: idf table

	cat	**dog**	**household**	**love**	**useful**
idf	0.2877	0	0.6931	0.2877	0.6931

Hence, we get the raw $tf \times idf$ values as:

Table 3: Raw $tf \times idf$ table – note: these values are referred to as $rt_{d\#}(j)$

	cat	**dog**	**household**	**love**	**useful**
d_1	0.5754	0	0	0.2877	0
d_2	0.5754	0	0	0	0
d_3	0	0	2.7724	0.2877	2.0793
d_4	2.3016	0	1.3862	1.7262	2.7724

Applying the functions given in problem, we can calculate that:

L_1-normalization terms for each document:

$$norm_1(d_1) = \sum_{j=1}^{5} rt_1(j) = 0.8631;$$

$$norm_1(d_2) = \sum_{j=1}^{5} rt_2(j) = 0.5754;$$

$$norm_1(d_3) = \sum_{j=1}^{5} rt_3(j) = 5.1394;$$

$$norm_1(d_4) = \sum_{j=1}^{5} rt_4(j) = 8.1864.$$

L_2-normalization terms for each document:

$$norm_2(d_1) = \sqrt{\sum_{j=1}^{5} [rt_1(j)]^2} = 0.6433;$$

$$norm_2(d_2) = \sqrt{\sum_{j=1}^{5} [rt_2(j)]^2} = 0.5754;$$

$$norm_2(d_3) = \sqrt{\sum_{j=1}^{5} [rt_3(j)]^2} = 3.4774;$$

$$norm_2(d_4) = \sqrt{\sum_{j=1}^{5} [rt_4(j)]^2} = 4.2291.$$

Pivot document length normalization terms for each document:

$p = (norm_2(d_1)+norm_2(d_2)+norm_2(d_3)+norm_2(d_4))/4 = 2.2313$,

$$norm'(d_1) = m' \times norm_2(d_1) + (1 - m')p = 0.2 \times 0.6433 + 0.8 \times 2.2313 = 1.9137;$$

$$norm'(d_2) = m' \times norm_2(d_2) + (1 - m')p = 0.2 \times 0.5754 + 0.8 \times 2.2313 = 1.9001;$$

$$norm\ (d_3) = m\ \times norm_2(d_3) + (1 - m\ )p = 0.2 \times 3.4774 + 0.8 \times 2.2313 = 2.4805;$$

$$norm\ (d_4) = m\ \times norm_2(d_4) + (1 - m\ )p = 0.2 \times 4.2291 + 0.8 \times 2.2313 = 2.6309$$

Problem Part 2:
Further compute these three documents' ranking according to the query given before, under L_2-normalization and pivot document length normalization, and compare the ranks.

Part 2 Solution:
Compute relevance by term-weight measure for each document:

$$score(d) = \sum_{j \neq 0} (q[j] \times \frac{tf_d[j] \times idf[j]}{norm(d)}),$$

but, given q as a binary query vector, the function can be simplified as:

$$score(d) = \frac{\sum_{j, q[j] \neq 0} (tf_d[j] \times idf[j])}{norm(d)}.$$

Under L_2-normalization:

$$score(d_1) = \frac{0.2877 + 0.5754}{0.6433} = 1.3417;$$

$$score(d_2) = \frac{0 + 0.5754}{0.5754} = 1;$$

$$score(d_3) = \frac{0.2877 + 0}{3.4774} = 0.0827;$$

$$score(d_4) = \frac{2.3016 + 1.7262}{4.2291} = 0.9524.$$

So the ranking under L_2-normalization is:

$$\boldsymbol{d_1 > d_2 > d_4 > d_3}$$

Under pivoted document length normalization:

$$score(d_1) = \frac{0.2877 + 0.5754}{1.9137} = 0.4510;$$

$$score(d_2) = \frac{0 + 0.5754}{1.9001} = 0.3028;$$

$$score(d_3) = \frac{0.2877 + 0}{2.4805} = 0.1160;$$

$$score(d_4) = \frac{2.3016 + 1.7262}{2.6309} = 1.5310.$$

So the ranking is adjusted by pivot document length normalization as:

$$\boldsymbol{d_4 > d_1 > d_2 > d_3}$$

We can see ranks for most relevant retrieved documents change among d_1, d_2 and d_4. Simple observation of table 1 can tell us that d_4 is very likely to be more relevant to the given query comparing to d_1 and d_2, since the query terms appear in d_4 very frequently. If you examine the values of normalization terms calculated above, you can see that the L_2-normalization terms fall into a larger range than pivoted document normalization. The long document, d_4 has a value almost 9-times larger than short document d_1. This verifies the claim made in [SBM'96] that L_2-normalization penalizes long documents too much, which harms the performance.

Problem Part 3:

As we demonstrated in the lecture, if we pick the pivot point *p* at another place, *m'* can be adjusted accordingly to maintain the ranking. To test this idea, let's set *p* as the median L_2-normalization term from all the documents in the given corpus. What would be a reasonable range of *m'* such that the ranking under pivoted document length normalization (of the L_2-normalization) for query "love cat" is still maintained?

Part 3 Solution:

The median normalization term given by L_2 is $(norm_2(d_1) + norm_2(d_3))/2$ = (0.6433+3.4774)/2 = 2.0604. So we set *p* = 2.0604.

We wish to find *m* such that $m' \in (0,1)$ and $score(d_1) \le score(d_4)$:

$$score(d_1) = \frac{rt_{d1}("love") + rt_{d1}("cat")}{m' \times norm_2(d_1) + (1-m')p} = \frac{0.2877 + 0.5754}{0.6433m' + 2.0604(1-m')}$$

$$\le score(d_4) = \frac{rt_{d4}("love") + rt_{d4}("cat")}{m' \times norm_2(d_1) + (1-m')p} = \frac{1.7262 + 2.3016}{4.2291m' + 2.0604(1-m')}$$

Solving, $m' \in (0, 0.8603)$**.**

Substituting m = 0.86:

$$score(d_1) = \frac{rt_{d1}("love") + rt_{d1}("cat")}{m' \times norm_2(d_1) + (1-m')p} = \frac{0.8631}{0.8417} = 1.0254$$

$$\le score(d_4) = \frac{rt_{d4}("love") + rt_{d4}("cat")}{m' \times norm_2(d_1) + (1-m')p} = \frac{2.3016}{2.0604} = 1.0261$$

INFO 630/CS 674 Lecture Notes: Deeper Thought

Scribes: Vladimir Barash, Stephen Purpura, Shaomei Wu

Our finger exercise demonstrated that the choice of normalization function impacted document ranking based on document length. But there is a fundamental difference between our finger exercise and the TREC data sets – limited vocabulary. Our finger exercise example and the TREC corpora are not degraded with misspellings or nonsense terms such as terms which might appear from a partially successful OCR (optical character recognition) process. In such a process, the term 'cat' may appear as 'cats', 'cal', or 'eat'.

Question:
Why might vocabulary diversity have an impact on the normalization function? Such vocabulary diversity might be caused by a significant presence of nonsense terms (misspellings, etc.).

Answer: We've already noted in the lecture that more non-zero term frequencies cause the retrieval process to be biased towards long documents. But we didn't examine the impact of extremes in vocabulary diversity on the effectiveness of the normalization function.

The finger exercise used 5 terms, while the TREC data is supposed to have a more "representative" vocabulary. Even the TREC corpora are a special case of real-world data because they are supposed to be error-free. Real-world vocabulary introduces two common problems for NLP researchers: synonymy and polysemy.

Unlike our finger example, real world corpora contain synonyms – like kitty and cat. Given a query which includes "cat", use of the term "kitty" in the document instead of "cat" reduces the term frequency of "cat" and can cause the information retrieval process to rank documents which contain the word "kitty" lower than other documents which might not even contain the word "cat" even though an expert reviewer would consider "kitty" as a synonym for "cat" and on-topic.

In addition, terms have a range of meanings based on the context. In American political documents, the word "choice" sometimes refers to the discussion of reproductive rights and other times it refers to an election. So, the query "choice abortion" might rank documents about elections higher than documents about reproductive rights if "choice" is not disambiguated.

In addition to vocabulary diversity caused by synonyms and polysemes, [SBM '96] mentions another reason that vocabulary diversity may exist – corpora may be degraded with nonsense terms and misspellings such as terms which might appear from a partially successful OCR (optical character recognition) process.

Now we'll examine an information retrieval example which is affected by these types of vocabulary diversity.

For the purposes of this exercise, we will use the following '*idf*' function (note this is different from the *idf* function used in the finger exercise but the same as the *idf* function used in the notes):

$$idf_d(j) = \left(\frac{N}{n_j} \right),$$

where N is the size of the corpus (the total number of documents in the corpus) and n_j is the number of documents that contain term j.

Compute relevance by term-weight measure for each document:

$$score(d) = \sum_{j \neq 0} (q[j] \times \frac{tf_d[j] \times idf[j]}{norm(d)}),$$

but, given q as a binary query vector, the function can be simplified as:

$$score(d) = \frac{\sum_{j, q[j] \neq 0} (tf_d[j] \times idf[j])}{norm(d)}.$$

Increasing the number of term synonyms in the vocabulary varies $tf_d[j]$ by *increasing the probability* that $tf_d[j]$ will decrease in value. Similarly, when polysemes are present, they will artificially increase $tf_d[j]$. When the $tf_d[j]$ terms increase or decrease in value *relative to the real match against the user's information goal*, the normalization function can play havoc on the retrieval probabilities.

Consider the following example documents in a 5 document corpus:

Documents:

Doc#1:
Cats are cool, soft, fuzzy and bouncy!

Doc#2:
Dogs love to eat and run around.

Doc#3:
It was raining cats and dogs the other night... so bad that I couldn't go outside. Sometimes I would come to the window and just stare at the rain. It was very depressing, but in the morning, I felt better!

Doc#4:
It's a dog-eat-dog world out there. From puppies to big hounds, everyone struggles to survive, to avoid his superior and to beat up on his inferior. That's just how it is.

Doc#5:
Cats and dogs are two common types of household animals. There are many species of cats and dogs - from the common house cat, to the Blue Russian, from bulldog to shepherd. Both cats and dogs have been domesticated by man many thousands of years ago and are loved and cared for by many pet owners today. There are even urban legends of cat owners having statistically better health than non-cat owners - and everyone knows how useful a dog can be, for protecting the house, for instance! There are many more things to say about cats and dogs, but I think I've run out of time, so I have to go. Thank you for listening!

If you run an experiment where you find document weights for both the full text vocabulary and for the Porter stemmed vocabulary (with stop word removal) versions, you will find very different match weightings. (For more information on Porter stemming, see http://tartarus.org/~martin/PorterStemmer/)

IMPORTANT NOTE: “non-cat” and “dog-eat-dog” are treated as single words by the Porter stemmer and distinct, separate words by the term frequency counting algorithm that we used to calculate word frequencies. See the References section for links to the complete output of the tools.

Now examine the results of the query: {“cat”, “love”}

Full vocabulary

Document #	raw tfidf squared for ‘cat’	raw tfidf squared for ‘love’
1	0	0
2	0	25.00
3	0	0
4	0	0
5	225.00	0

Porter stemmed vocabulary

Document #	raw tfidf squared for ‘cat’	raw tfidf squared for ‘love’
1	2.78	0
2	0	6.25
3	2.78	0
4	0	0
5	100.00	6.25

Full vocabulary

Document #	L_2-Norm	Pivoted TFIDF Norm
1	10.4894	23.7915
2	8.2365	23.3409
3	28.3702	27.3677
4	25.6369	26.8210
5	62.8521	34.2640

Porter stemmed vocabulary

Document #	L_2-Norm	Pivoted TFIDF Norm
1	10.1379	17.4178
2	6.3465	16.6595
3	19.1848	19.2272
4	17.5000	18.8902
5	43.0200	23.9942

Full vocabulary

Document #	L_2-Normed Score	Pivoted TFIDF Normed Score
1	0	0
2	0.6071	0.2142
3	0	0
4	0	0
5	0.2387	0.4378
Rank:	$\mathbf{d_2 > d_5 > d_1, d_3, d_4}$	$\mathbf{d_5 > d_2 > d_1, d_3, d_4}$

Porter stemmed vocabulary

Document #	L_2-Normed Score	Pivoted TFIDF Normed Score
1	0.1644	0.0957
2	0.3939	0.1501
3	0.0869	0.0867
4	0	0
5	0.2906	0.5210
Rank:	$\mathbf{d_2 > d_5 > d_1 > d_3 > d_4}$	$\mathbf{d_5 > d_2 > d_1 > d_3 > d_4}$

For either L_2-Normalization weightings or Pivoted TFIDF weightings, ***the Porter Stemmed result produces a better document ranking than the full vocabulary versions***. But in this case, the Pivoted TFIDF scores always outperform the L_2-Normalized scores for both the full vocabulary and the Porter stemmed vocabulary. In the documents, there are no synonyms for 'cat' or 'love', although 'pets' could be considered a synonym for 'cat' and "non-cat" causes some noise.

Now consider the query: {“dog”, “love”}

Full vocabulary

Document #	raw tfidf squared for ‘dog’	raw tfidf squared for ‘love’
1	0	0
2	0	25.00
3	0	0
4	25.00	0
5	6.25	0

Porter stemmed vocabulary

Document #	raw tfidf squared for ‘dog’	raw tfidf squared for ‘love’
1	0	0
2	2.78	6.25
3	2.78	0
4	0	0
5	69.44	6.25

Full vocabulary

Document #	L_2-Normed Score	Pivoted TFIDF Score
1	0	0
2	0.6071	0.2142
3	0	0
4	0.1950	0.1864
5	0.0398	0.0730
Rank:	**$d_2 > d_4 > d_5 > d_1, d_3$**	**$d_2 > d_4 > d_5 > d_1, d_3$**

Porter stemmed vocabulary

Document #	L_2-Normed Score	Pivoted TFIDF Score
1	0	0
2	0.6565	0.2501
3	0.0869	0.0867
4	0	0
5	0.2518	0.4515
Rank:	**$d_2 > d_5 > d_3 > d_1, d_4$**	**$d_5 > d_2 > d_3 > d_1, d_4$**

Like the “cat love” query, for the “dog love” query, Pivoted TFIDF scores outperform L_2-Normalized scores on the Porter stemmed vocabulary. Within the Porter stemmed vocabulary, L_2-Normalized scores prefer short documents and Pivoted TFIDF scores prefer long documents. But the Pivoted TFIDF scores fail to outperform L_2-Normalized scores on the full vocabulary. Most of the reason for this is the presence/absence of the query terms in the document. In d_4, ‘dog’ has a polyseme – ‘dog-eat-dog’ which is filtered as a different term by the Porter stemming algorithm. However, in the full vocabulary, two occurences of ‘dog’ in ‘dog-eat-dog’ cause d_4 to erroneously be ranked as highly relevant and affect the normalization score.

How much is the normalization score affected? You can think of the normalization factor as an amplifier of the 'tf idf' terms. Documents are being rewarded or punished due to the type of normalization. The following table shows how much more the Pivoted TFIDF Normed scores are being punished under the full vocabulary, compared to the Porter stemmed vocabulary. When the numbers in the table are larger, the L_2-Norm is comparatively greater than the Pivoted TFIDF Norm.

Ratio of L_2-Norm/Pivoted TFIDF Norm

Document #	Full vocabulary	Porter stemmed vocabulary
1	0.4409	0.5820
2	0.3529	0.3810
3	1.0366	0.9978
4	0.9559	0.9264
5	1.8343	1.7930

Combining the effects of the vocabulary diversity (flatter or inflated 'tf's and different norm scores), it is easy to see how retrieval probabilities can change based on the vocabulary diversity.

References:

See http://docs.google.com/Doc?id=dcpkz9gb_42wmz5b5 for the full texts, processing instructions, and raw statistics about the text.

For a spreadsheet of the full vocabulary document matrix and statistics, see:
http://spreadsheets.google.com/pub?key=pswp60NXd6HBLztSVi-eGcw

For a spreadsheet of the Porter stemmed vocabulary document matrix and statistics, see:
http://spreadsheets.google.com/pub?key=pswp60NXd6HAkLlHrLiEtEg