CS630 Lecture Notes

Lecturer: Lillian Lee
Scribes: Chris Danis (cgd3) & Brian Rogan (bcr6)

Lecture 2: 31 January 2006

1 Review & Introduction

Last time we spoke about the vector space model. Today we will be covering recent research on the VSM. In particular, we'll begin discussing the following paper: "Pivoted document length normalization", by Singhal, Buckley, Mitra, from SIGIR '96 [1]. These three were all grad students at Cornell.

Why are we talking about this?

- It's a state-of-the-art VSM term-weighting method. Still widely used; Google Scholar lists 263 citations.
- It's an interesting example of empirically-driven research. There's no underlying theory expressing a grand scheme of how to represent information or determine relevance – the aim is simply to get the best performance.
- Also, it's "cute": a little math based on a simple but powerful observation that produces good results in practice.

1.1 Vector space model: review

A reminder of what we're trying to do, and of notation: For a given document d, we're trying to create a document vector

$$\vec{d} = \begin{bmatrix} d_1 \\ \vdots \\ d_m \end{bmatrix} = (d_1, \cdots, d_m)^T$$

with each element of the vector corresponding to a term from the vocabulary $v^{(1)}, \cdots, v^{(m)}$.

What is it we think ought to be included in a term weight in our document vector? There's consensus amongst researchers that a term weight should include the following:

1. "idf_j": inverse document frequency. A function that decreases in the fraction of the documents in a corpus that contain that term. Basically, the more common a term is in a corpus, the less distinguishing (and thus less important) it is. There are many possible specific functions; common ones include things like $\log\left(\frac{\#\text{ of documents}}{\#\text{ of documents containing term}}\right)$. Note that there is an idf_j for each term and corpus, and *not* for each specific document.
2. "$tf_j(d)$": term frequency. A function that increases in the number of times the term $v^{(j)}$ appears in the document d. This is obviously important; documents containing a high frequency of terms similar to the terms given in the query are probably relevant. However, looking just at the raw number can be misleading: consider, for instance, search engine spam. Repetition can artificially inflate the ranking of a document. So...
3. Normalization. "$\text{norm}_j(d)$" compensates for the inherent bias of raw frequencies towards long documents. The assumption that the bias towards long documents is a bad thing to be countered is an interesting one. For now we will make this assumption, but we question it later. We explain why the bias exists in the next section.

So, given a document and a query, the document will be scored by the function:

$$\vec{q} \cdot \vec{d} = \sum_j q_j \times idf_j \times \frac{tf_j(d)}{norm_j(d)}$$

2 Normalization & long documents

2.1 The problem

There are two reasons why that, without normalization, long documents are favored for retrieval.

Long documents often use many of the same terms repeatedly, leading to higher tf_j terms, and thus, higher scores on certain queries. Longer documents also have many different terms (simply by virtue of having more words), and thus, many different non-zero tf_j terms can also boost scores, even if they are individually not that significant.

What we hope is to have a normalization function that compensates for this length bias.

2.2 Solutions

A very common solution is this function. Note it is term-independent but document-dependent:

$$norm_j(d) = norm(d) = \sqrt{\sum_j tf_j(d)^2}$$

This is just the 2-norm for $\vec{d}$. This sets $||\vec{d}||_2 = 1$.

Recall our usual ranking function for searches in the VSM:

$$\vec{q} \cdot \vec{d} = ||\vec{q}||_2 ||\vec{d}||_2 \cos(\angle(\vec{q}, \vec{d}))$$

For ranking documents over one query, we can drop the $||\vec{q}||_2$ term, as it is the same in each score. And, if we normalize with the 2-norm function, we can also drop $||\vec{d}||_2$, as it will just be 1. Thus, for ranking purposes, the function is equivalent to

$$\vec{q} \cdot \vec{d} \stackrel{\text{rank}}{=} \cos(\angle(\vec{q}, \vec{d}))$$

Now we're just looking at directionality and not at dogmaticity. Is this really a benefit? It's clean, it's easy to compute, but is it the best we can do?

Why not set the norm to $norm(d) = \max_j tf_j(d)$? It's a very simple idea; used a lot in practice. The problem is outliers (one very common word in a document will squash down all the other term weights). It also ignores most of the distribution of terms (consider a document containing each word ten times, versus a document containing one word ten times and the other once).

After we've done the 2-norm normalization, the document vector length is completely ignored. All the vectors have the same length, so how can length have any effect? This is exactly what we want for good retrieval performance if there's no correlation between length and relevance. Singhal et al. examine these assumptions.

An aside: IR people worry about overfitting much less than machine learning people do (although it is still a concern). Our argument is as follows: in the end, we have one single document collection of interest: the Web. So what if we have to retune our system to perform well on "the Martian web"?

3 Singhal, Buckley, and Mitra

Data from TREC was used in all of the experiments. We can think of this data as looking like the following:

$$q^{(1)} \quad \rightarrow \quad d^{(14)}, d^{(20)} \ldots$$

$$\vdots$$

$$q^{(50)} \rightarrow d^{(15)}, d^{(400)}, \ldots$$

Each query has a list of documents relevant to it. We denote the set of relevant documents as Rel. In this case, $|\text{Rel}| = 9805$. These examples were generated by pooling results of "good" IR systems, and were then hand-checked for relevance[1].

Let's consider a plot of

$$\frac{\#(d \in \text{Rel of a given length})}{|\text{Rel}|}$$

[1] Thus Rel contains only relevant documents (i.e. no false positives), but may not contain all relevant documents (there may exist false negatives).

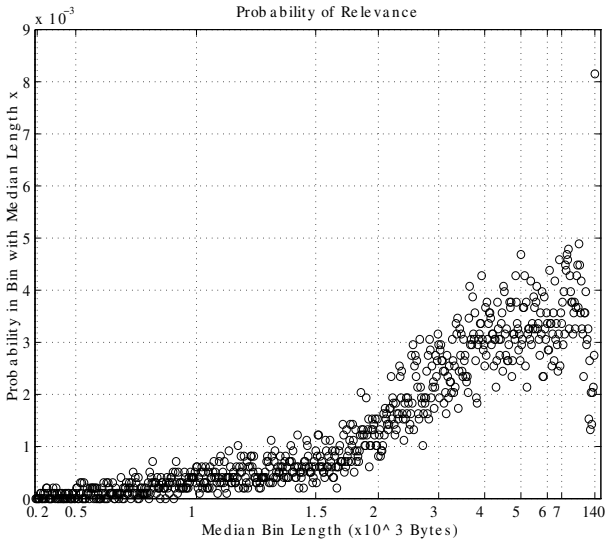

Figure 1: Probability of relevance versus length, reproduced from [1].

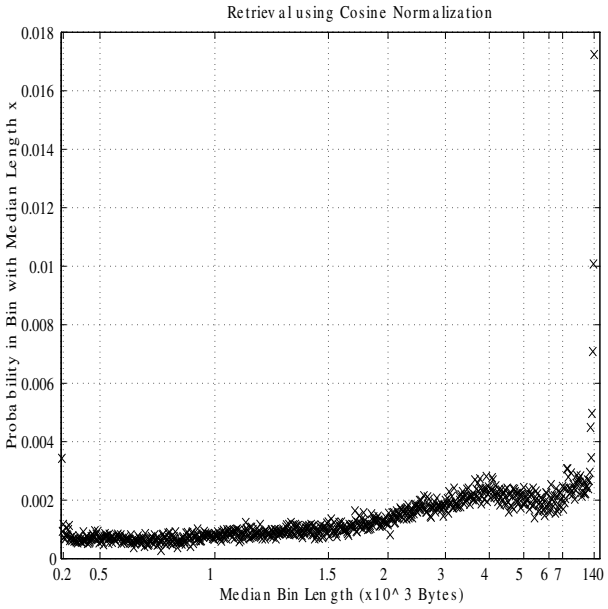

Figure 2: Retrieval using cosine normalization, also reproduced from [1].

versus length. Singhal et al. perform this experiment in their paper, the result is shown in Figure 1.

Essentially we're looking at how the relevant documents fall with respect to length. If our assumption that length and relevance are independent is correct, we should see something like a flat line. But we do not! Please remember that this does *not* say that "if a document is longer it's more likely to be relevant" but rather the converse: "if a document is relevant it's likely to be longer".

But we also need to consider: does the cosine actually ignore document length? Using the TREC data, we run through each query and return the top 1000 results, using a standard cosine retrieval system. Call this set Ret. Plot

$$\frac{\#(d \in \text{Ret of a given length})}{|\text{Ret}|}.$$

This plot is shown in Figure 2. Surprisingly, the plot shows a positive correlation between length and probability of retrieval. We need to explain this result[2]. Our 2-norm normalization function does account for document-length effects – it grows as the number of terms and the size of term frequencies increase – but it doesn't completely destroy the length information.

If we plot these two curves against one another, what we'd like is for them to mostly line up. Such a result would indicate that we are "fairly" retrieving documents of all lengths, and thus, the distribution of lengths of documents retrieved should be rather close to the distribution of lengths of relevant documents. Singhal et al. performed this experiment; the result is reproduced in Figure 3.[3]

Here we have an interesting result. Cosine retrieval

[2]Extended discussion in class indicated that it was unlikely that this bias was solely a result of the pooling technique used to select the Rel set. For more commentary and analysis on this topic, see [2].

[3]Singhal et al. constructed their graphs by first sorting documents (either relevant docs or retrieved docs, depending on length). They then divided that set by making bins of 1000 documents each. x-points are the median length for each bin (actual data points not necessarily spaced evenly). The advantage is that, for each data point, we have 1000 samples – increases confidence of our statistics. This procedure was used for Figure 1 and for Figure 2. For Figures 3 and 4, further smoothing was performed in addition to the binning.

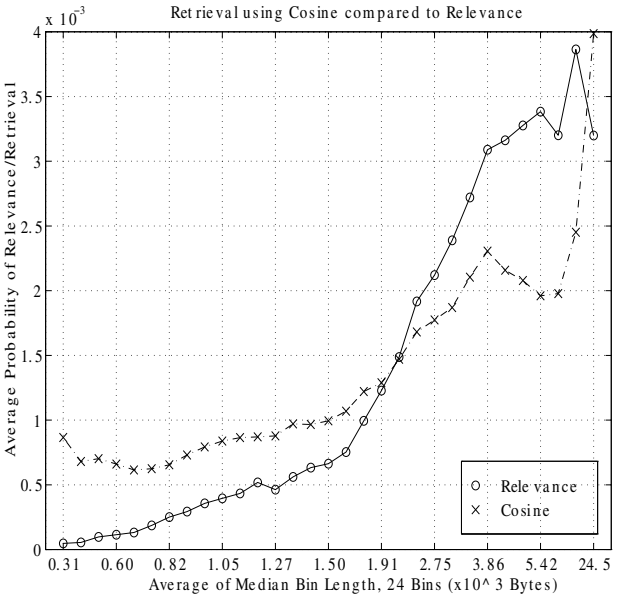

Figure 3: Retrieval using cosine compared to relevance; also reproduced from [1].

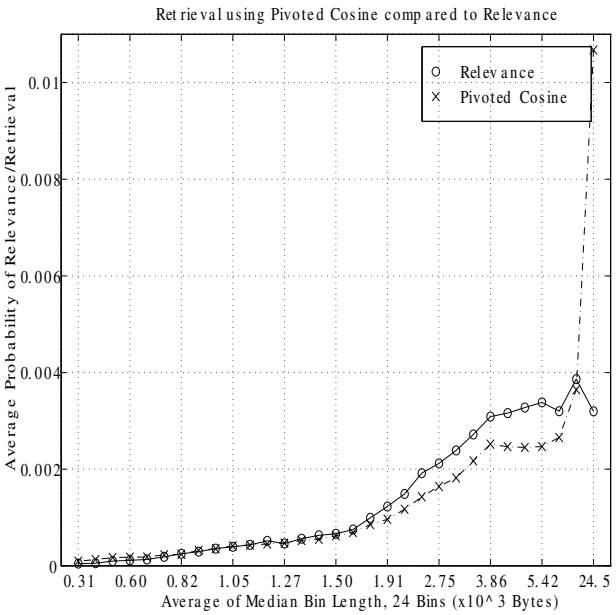

Figure 4: Retrieval using pivoted cosine compared to relevance (reproduced from [1]).

tends to retrieve short documents more often than those short documents are relevant, and to retrieve long documents less often than those documents are relevant. Ideally the two curves would match.

We apply a simple technique to adjust the retrieval curves. Consider the following: we can define a new normalization function based on our old one:

$$norm'(d) = m' \times norm(d) + b'$$

This is just a linear function of our old normalization function. Next lecture we will discuss how to select a "pivot point" and set parameters such that the retrieval curve can be made to agree with the relevance curve; see Figure 4 for a "sneak peek" at the results. Choosing parameters correctly leads to a great increase in precision.

References

[1] Amit Singhal, Chris Buckley, and Mandar Mitra. Pivoted document length normalization. In *Research and Development in Information Retrieval*, pages 21–29, 1996.

[2] Justin Zobel. How reliable are the results of large-scale information retrieval experiments? In *SIGIR*, pages 307–314. ACM, 1998.