CMU Spoken Document Retrieval in Trec-8:
Analysis of the role of Term Frequency TF

M. Siegler, R. Jin and A. Hauptmann

The participation of Carnegie Mellon University in the TREC-8 Spoken Document Retrieval Track used the basic same Sphinx speech recognition system as in TREC-7. Due to some unfortunate defaults in the parameter setup files, the speech recognizer did not perform in a reasonable manner. We will not analyze the results of the speech recognizer runs, as we believe the results contained abnormal types of errors, and insights or improvements on these errors would not generalize. A thorough examination of the speech recognition condition is given in [3]. However, we did evaluate a slightly modified weighting scheme in the reference (R1) and baseline (B1) conditions, which is described below.

1. Motivation for the Retrieval Formulas

The formula we used for cmu-r1 and cmu-b1 runs is based on dtb described by Singhal[1]. In addition, to improve the accuracy, we use the standard pseudo-relevance feedback [2], conservative collection enrichment [1] and document expansion [1]. Since what we have done in that part is similar to what AT&T has done in TREC7 [1], we are not repeating the description of this approach.

Here we would like to discuss the modification we make in the dtb formula. As many people pointed out in previous TRECs, directly multiplying term frequency tf with inverse document frequency idf generally causes poor performance. The poor performance seems to be due to an overweighting of the tf term. To avoid this overestimation of tf, researchers have used ln(tf+1) or even ln(ln(tf)+1)+1.
However, these approaches look more empirical than theoretical. Our modification on the tf term is a theoretically motivated attempt to resolve this problem.

2. Analysis

Usually idf for a word A is written as log((N+1)/M). Here N is the total number of documents in the collection and M is number of documents having at least one occurrence of the word A within the collection. In other words, idf for word A can be thought of as -log(p) where p is the probability that a document contains at least one occurrence of word A in the collection. Then the term tf*idf for a word A can be written as

Tf * idf = -tf * log(p) = -log(ptf). (1)

We can easily extend the meaning of idf to interpret the term tf*idf for word A as -log(p’) and p’ is the probability that a document contains at least tf occurrences of word A. So we have

Tf * idf = -log(p) (2)

Combing the two formulas together, we have
p’ = ptf,

which means that the probability that a document contains at least tf occurrences of the word A is the probability that a document contains at least one occurrence of the word A to the power tf. Obviously this can be true only when the independent assumption that one occurrence of the word A has nothing to do with another occurrence of the word A is correct.
However, because of the complicated correlation within word occurrences, the independent assumption generally is wrong and will cause underestimation of probability p'. We think this is the reason why multiplying tf with idf directly generally causes overestimation and gives rise to poor performance.

3. Solution

To avoid this problem of overestimation by multiplying tf directly, we have come up with two solutions to replace tf*idf. Since tf*idf for word A can be interpreted as -log(p') and p' is the probability that a document contains at least tf occurrences of word A, the key issue is how to estimate this probability p'.

One solution is using the word histogram directly. For each word A, we can build up a histogram function N(x, A) that tells the number of documents containing exact x occurrences of the word A. With this histogram function, we can compute the "tf*idf" for word A as
log((N+1)/G(tf, A)).
Here G(tf, A) is defined as

G(tf, A) = Sum(N(x, A)) over x and x is integer from x to infinity.

The second method uses a fitted Gaussian distribution to estimate the probability p'. For each word A, we can compute the average occurrences of word A avg_A and standard deviation of occurrences of word A std_dev_A from the histogram function N(x, A). Now we can build the normalized Gaussian distribution as D(x, avg_A, std_dev_A).
Then the tf*idf" can be computed as
-log(I(tf, A)).
Here I(tf, A) is defined as

I(tf, A) = Integral of D(x, avg_A, std_dev_A) over x and x is from tf to infinity.

Furthermore we can use the standard error function to represent I(tf, A) as the following:

0.5 * err((tf - avg_A)/sqrt(2)/std_dev_A)
if tf >= avg_A
I(tf, A) =
0.5 * err((avg_A - tf)/sqrt(2)/std_dev_A) + 0.5
if tf < avg_A

Both these two approaches have their advantages and disadvantages. The good side of first approach is that it uses the exact data and makes no assumption or approximation. However it may be misled by the local fluctuation. As for the second approach, it complements the down side of the first approach by using fitted Gaussian distribution. However it may cause disaster if the data doesn't fit in Gaussian distribution or when a small data set doesn't reliably estimate the true average and standard deviation.

To obtain the good properties of both approaches, we created a new method. We modify the histogram function N(x, A) for word A as follows: Instead of using natural granularity 1 for x, we use standard deviation std_dev_A as the granularity. We define the granularity function for each word A as

grand(A) = MAX(std_dev_A/3, 1).

Now we can define a new histogram function N'(x, A) as

N'(x, A) = Sum(N(y, A)) over y and y is from floor(x / grand(A))*grand(A) to ceiling(x / grand(A))*grand(A).

From the definition of N'(x, A), it is easily seen that N'(x, A) is the same for all x in the range [floor(x / grand(A))*grand(A) .. ceiling(x / grand(A))*grand(A)].

Now we can use the first approach to compute the "tf*idf" except that this time the histogram function is N'(x, A) instead of N(x, A). Since the new histogram function is defined as a sum of the old histogram function over an interval on the order of standard deviation, it will be more stable and avoids some risks of the first approach.

4. Experiment

From experiments we have performed on the TREC data, we find out that the approach described above for computing factor "tf*idf" is better than tf * idf or (ln(tf) + 1) * idf. However, we did not see any significant improvement in performance of our formula over (ln(ln(tf) + 1) +1) * idf. Instead our formula is generally slightly worse than the (ln(ln(tf) + 1) + 1) * idf factor.

By comparing documents weighted by our schema and weighted by (ln(ln(tf) + 1) + 1) * idf, we find out that our schema still has the problem of overestimating tf especially when the tf is larger. We think it is due to the fact that when tf is close to the largest tf, the G(tf, A) is very inaccurate because of the lack of histogram data. In the future we will introduce special treatment for this "ending effect".

5. Conclusion

In this paper we attempted tf*idf a more theoretic interpretation and point out a possible reason why multiplying tf directly with idf causes the poor performance with the given interpretation. We came up with a word histogram based method of integrating tf and idf into one factor which is log(1/p') and p' is the probability that a document has at least tf occurrences of a particular word. We have several success over tf*idf and (ln(tf)+1)*idf and fail to compete with (ln(ln(tf)+1)+1)*idf. We think the failure is due to the "ending effect" and we will pursue the problem further in the future.

6. References

[1] Singhal.A. AT&T at TREC7. In E. M. Voorhees and D. K. Harman, editors, TEXT RETRIEVAL CONFERENCE (TREC-7), page 141-151, 1998.

[2] J.J Rocchio. Relevance feedback in information retrieval. In The SMART Retrieval System-Experiments in Automatic Document Processing, page 313-323, Englewood Cliffs, NJ, 1971. Prentice Hall, Inc.

[3] Siegler,M. Integration of Continuous Speech Recognition and Information Retrieval for Mutually Optimal Performance, PhD. Thesis, Electrical and Computer Engineering, Carnegie Mellon University, 1999
http://www.cs.cmu.edu/~msiegler/publish/PhD/thesis.ps.gz