Notes on the Lemur TFIDF model

Chengxiang Zhai
School of Computer Science
Carnegie Mellon University

October 22, 2001

1 Introduction

The TFIDF model is often used as a baseline model for comparison with new retrieval models. However, it is actually not a well-defined model, in the sense that there are several heuristic components in the model that can affect the performance significantly.

In the Lemur toolkit, we implemented one variant of the TFIDF model family that is relatively robust and performing well empirically. The model is based on the Okapi TF formula originally derived from a probabilistic model[1]. In this notes, we describe the details of the model and its implementation.

2 Simple Retrieval Model

We assume each document and each query are represented by a term frequency vector $\vec{d} = (x_1, x_2, ..., x_n)$ and $\vec{q} = (y_1, y_2, ..., y_n)$ respectively, where n is the total number of terms, or the size of vocabulary and x_i, y_i are the frequency (i.e., the counts) of term t_i in d and q respectively. Given a collection C, the well-known inverse-document-frequency (idf) of a term t is given by $\log(N/n_t)$, where N is the total number of documents in C and n_t is the number of documents with term t. All terms in a query or a document are weighted by the heuristic TF-IDF weighting formula. That is, the weighted vectors for $\vec{d}$ and $\vec{q}$ are:

$$\begin{aligned} \vec{d}' &= (tf_d(x_1)idf(t_1), tf_d(x_2)idf(t_2), ..., tf_d(x_n)idf(t_n)) \\ \vec{q}' &= (tf_q(y_1)idf(t_1), tf_q(y_2)idf(t_2), ..., tf_q(y_n)idf(t_n)) \end{aligned}$$

The document TF function tf_d is given by the Okapi TF formula with parameters k_1 and b:
$tf_d(x) = \frac{k_1 x}{x + k_1(1 - b + b\frac{l_d}{l_C})}$

The query TF function tf_q is defined similarly with a parameter l_Q representing average query length[1].

The score of document $\vec{d}$ against query $\vec{q}$ is given by
$s(\vec{d}, \vec{q}) = \sum_{i=1}^{n} tf_d(x_i) tf_q(y_i) idf(t_i)^2$

There are five parameters to set: k_1 and b for the document TF function and k_1, b, l_Q for query TF. Robertson et al. [2] suggested a default value of 1.2 for k_1 and 0.75 for b for document TF, but the change of k_1 and b may affect the performance, sometimes significantly. We usually set b to 0 and k_1 to $1,000$ for query TF, which makes our query TF formula almost identical to the original BM25 query TF formula where k_3 is our k_1. Sometimes, allowing the b for query to take a non-zero value is beneficial. In that case, k_1 and b would be set in the same way as they are set for a document. l_Q is usually set to 3 for title queries.

3 Feedback Retrieval Model

If we know that some documents are relevant to a query and some are not, we may be able to learn from these example documents to improve a query. This is called feedback in retrieval. There are two different scenarios for feedback: relevance feedback and blind/pseudo feedback, but they only differ in the way to identify the relevant/non-relevant document subset and their underlying feedback mechanism is the same. Relevance feedback refers to the scenario where a user judges the results from any previous retrieval and tells the system which documents are relevant and which are not, whereas in blind feedback, the top M documents from any previous retrieval results are assumed to be relevant and all others or some of others are assumed to be non-relevant.

In the Lemur toolkit, we implemented a simplified Rocchio feedback algorithm. The basic idea of the Rocchio algorithm is as follows. Let R and $\bar{R}$ be the relevant document set and non-relevant document set, respectively, to be used for feedback. The Rocchio algorithm simply "moves" the query vector closer to the centroid vector of R and away from the centroid vector of $\bar{R}$. Formally,

[1]This is different from the original query TF formula implied by the BM25 retrieval function, which does not have the parameter of average query length, but the difference is insignificant for short queries.

$\vec{q}_{new} = \vec{q}_{old} + \alpha\vec{d}_R - \beta\vec{d}_{\bar{R}}$

where, $\vec{d}_R$ is the centroid vector (i.e., the average) of all weighted document vectors in R, and $\vec{d}_{\bar{R}}$ that in $\bar{R}$.

In our implementation, we made two simplifications which tend not to affect the performance so much empirically.

(1) We ignored $\bar{R}$, i.e., $\beta = 0$. (2) We truncated $\vec{d}_R$ so that only the top-weighted K terms are kept and the rest are set to zero.

Therefore, when used for blind feedback, our implementation of Rocchio has three parameters to set: M, K, and α. They are all set empirically. We usually allow M and K to take a value from 10 to 100, and α to take a value in $(0, 4]$.

References

[1] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford (1995). "Okapi at TREC-3," in D. K. Harman (ed), *The Third Text REtrieval Conference (TREC-3)*, NIST Special Publication.

[2] S. E. Robertson, and S. Walker (2000), "Okapi/Keenbow at TREC-8," in E. Voorhees and D. K. Harman (editors), *The Eighth Text REtrieval Conference (TREC-8)*, NIST Special Publication 500-246.