Technical Report

UCAM-CL-TR-356
ISSN 1476-2986

Number 356

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Simple, proven approaches to text retrieval

S.E. Robertson, K. Spärck Jones

December 1994

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

This version of the report incorporates minor changes to the December 1994 original, which were released May 1996, May 1997 and February 2006.

Technical reports published by the University of Cambridge Computer Laboratory are freely available via the Internet:

http://www.cl.cam.ac.uk/TechReports/

ISSN 1476-2986

Simple, proven approaches to text retrieval

S.E. Robertson and K. Sparck Jones
Microsoft Research Ltd, Cambridge and Computer Laboratory, University of Cambridge

December 1994 version with updates May 1996, May 1997, February 2006

Abstract

This technical note describes straightforward techniques for document indexing and retrieval that have been solidly established through extensive testing and are easy to apply. They are useful for many different types of text material, are viable for very large files, and have the advantage that they do not require special skills or training for searching, but are easy for end users.

The document and text retrieval methods described here have a sound theoretical basis, are well established by extensive testing, and the ideas involved are now implemented in some commercial retrieval systems. Testing in the last few years has, in particular, shown that the methods presented here work very well with full texts, not only title and abstracts, and with large files of texts containing three quarters of a million documents. These tests, the TREC Tests (see Harman 1993–1997; IP&M 1995), have been rigorous comparative evaluations involving many different approaches to information retrieval.

These techniques depend on the use of simple terms for indexing both request and document texts; on term weighting exploiting statistical information about term occurrences; on scoring for request-document matching, using these weights, to obtain a ranked search output; and on relevance feedback to modify request weights or term sets in iterative searching.

The normal implementation is via an inverted file organisation using a term list with linked document identifiers, plus counting data, and pointers to the actual texts.

The user's request can be a word list, phrases, sentences or extended text.

1 Terms and matching

Index terms are normally content words (but see section 6). In request processing, stop words (e.g. prepositions and conjunctions) are eliminated via a stop word list, and they are usually removed, for economy reasons, in inverted file construction.

Terms are also generally stems (or roots) rather than full words, since this means that matches are not missed through trivial word variation, as with singular/plural forms. Stemming can be achieved most simply by the user truncating his request words, to match any inverted index words that include them; but it is a better strategy to truncate using a standard stemming algorithm and suffix list (see Porter 1980), which is nicer for the user and reduces the inverted term list.

The request is taken as an unstructured list of terms. If the terms are unweighted, output could be ranked by the number of matching terms – i.e. for a request with 5 terms first by documents with all 5, then by documents with any 4, etc. However, performance may be improved considerably by giving a weight to each term (or each term-document combination). In this case, output is ranked by sum of weights (see below).

2 Weights

The idea behind term weighting is selectivity: what makes a term a good one is whether it can pick any of the few relevant documents from the many non-relevant ones.

There are three different sources of weighting data:

Collection frequency

Terms that occur in only a few documents are often more valuable than ones that occur in many. Collection frequency weights (also known as inverse document frequency weights) are defined as follows, for term `t(i)`:

Given

```
n  =  the number of documents term t(i) occurs in
N  =  the number of documents in the collection
```

the Collection Frequency Weight for a term is then

```
(1)   CFW (i)  =  log N - log n
```

(Implementationally, term collection frequencies are simply counts attached to the inverted file, and weights are computed for request terms as needed. The logarithm may be taken to any convenient base: one implementation uses as base two to the power 0.1, which ensures that all weights can be well approximated by integers in the range –32K to +32K. Note that the above formula can only give positive weights.)

Term frequency

The second source of weighting is a term's within-document frequency: the more often a term occurs in a document, the more likely it is to be important for that document. Thus while a term's collection frequency is the same for any document, its document frequency varies. The term frequency for term `t(i)` in document `d(j)` is:

```
TF (i,j)  =  the number of occurrences of term t(i) in document d(j)
```

(In practice, these counts are recorded along with document identifiers in the inverted file.)

Term frequency should not, however, be used just as it stands as a weighting factor, but must be related to the remaining source of information about documents, as follows.

Document length

The third input to weighting is the length of a document. A term that occurs the same number of times in a short document and in a long one is likely to be more valuable for the former. We therefore have the length of a document `d(j)` thus:

```
DL (j)  =  the total of term occurrences in document d(j)
```

The use of document length described below actually normalizes the measure by the length of an average document:

```
NDL (j)  =  (DL (j)) / (Average DL for all documents)
```

This has the advantage that the units in which `DL` is counted do not matter much. A very simple measure such as number of characters in `d(j)` can be quite adequate as a substitute for number of term occurrences.

Combining the evidence

The three kinds of data for each term need to be combined together and with those for other terms from the request, to give a matching score for the particular document against the request. There are various formulae for this combination: the one presented here (often referred to as BM25) has proved effective in trials during the TREC Programme (see Robertson *et al.* 1995).

For one term `t(i)` and one document `d(j)`, the Combined Weight is

```
(2)   CW (i,j)  =  [ CFW (i) * TF (i,j) * (K1+1) ] /

                   [ K1 * ( (1-b) + (b * (NDL (j)) ) ) + TF (i,j) ]
```

`K1` and `b` are tuning constants (see below). The formula ensures that (a) the effect of term frequency is not too strong (doubling `TF` does not double the weight), and (b) for a term occurring once in a document of average length, the weight is just `CFW`.

The overall score for a document `d(j)` is simply the sum of the weights of the query terms present in the document. Documents are ranked in descending order of their scores, for presentation to the user. (The system may have some cut-off value such as a maximum number of documents to include or a minimum score.)

The tuning constant `K1` modifies the extent of the influence of term frequency. Ideally, it should be set after systematic trials on the particular collection of documents. In the TREC Programme tests, with a somewhat heterogeneous collection of full-text news items, some abstracts, and some very long government reports, the value `K1=2` was found to be effective; it is probably a safe value to start on. (Higher values would increase the influence of `TF`; `K1=0` eliminates the influence altogether.) The constant `b`, which ranges between 0 and 1, modifies the effect of document length.[1] If `b=1` the assumption is that documents are long simply because they are repetitive, while if `b=0` the assumption is that they are long because they are multitopic. Thus setting `b` towards 1, e.g. `b=.75`, will reduce the effect of term frequency on the ground that it is primarily attributable to verbosity. If `b=0` there is no length adjustment effect, so greater length counts for more, on the assumption that it is not predominantly attributable to verbosity. We have found (in TREC) that setting `b=.75` is helpful.

Using collection frequency weights alone is appropriate for document retrieval where only e.g. titles or short texts are available for searching; the other components of weighting are pertinent when searching is on full texts. However even here collection frequency weighting alone may be use with some advantage, if the other information is not available.

3 Iterative searching

This is a natural development of searching using collection frequency weights as defined above. In it an initial search is used to obtain some documents that are assessed for their relevance to the user's request, and marked as relevant or non-relevant. The information thus obtained can be used either just to reweight the initial search terms, or to modify the original query by adding new terms. Changing the request term weights alone is often called relevance weighting; changing the request composition, typically by adding more terms, is usually called query expansion: in the process, original terms are reweighted.

[1] We use `b` rather than the more natural `K2` as constant name for compatibility with other publications, where `b` is used for this purpose and `K2` for something else.

Relevance weights

The basis for relevance weighting is simply the relation between the relevant and non-relevant document distributions for a search term modulated by its collection frequency, since a term could be in relevant documents just because it is in a lot of documents (collection frequency weights are indeed a special case of relevance weights). It is also necessary to allow for the fact that one is predicting matching behaviour on rather little relevance data, so one should in particular not assume that because a term has not been in any relevant documents so far, it never will be. Relevance weights are therefore estimates, which accounts for the 0.5s in the formula below.

Given, for term `t(i)`,

```
r  =  the number of known relevant documents term t(i) occurs in
R  =  the number of known relevant document for a request
```

the Relevance Weight is

```
(3)   RW (i)  =  log [ ( (r+0.5)(N-n-R+r+0.5) ) / ( (n-r+0.5)(R-r+0.5) ) ]
```

This formula can be used instead of `CFW` (formula (1)) for all terms used in a second or subsequent iteration. It can also be used for the first iteration by setting `r=R=0`; the resulting formula is normally a very close approximation to CFW as in (1). Note, however, that this approximation does not work with very frequent terms (such as some terms that might usually be on a stopword list). A term that occurs in most documents in the collection will be given a very small positive weight by (1), but a large negative weight by the approximation ((3) with `r=R=0`). A rough and ready strategy for avoiding this difficulty is simply to force negative weights to a small positive value. Some such strategy is probably a good safeguard in any case, but should certainly be used if no stoplist is applied. (The base for the logarithm may be the same as for `CFW` (see above).)

Query expansion

The power of relevance feedback comes not so much from reweighting the original query terms, as from expanding the query by adding new search terms to it. Essentially, terms may be taken from the documents assessed as relevant; however, some selection should be performed (it is not usually desirable to include all such terms).

Different combinations of user and system effort may be used to select new terms. A simple, effective, entirely automatic procedure is as follows. All terms taken from relevant documents are ranked according to their Offer Weight (see Robertson 1990):

```
(4)   OW (i) = r * RW (i)
```

Then the top 10 or 20 ranked terms are included in the search. 10 or 20 is probably fairly safe; in most cases it will include some rubbish terms, but the effect of these will be outweighed by the good ones.

A selection procedure including the user might involve showing him/her the terms in offer weight order and inviting selection or rejection on each one. Alternatively or additionally, the user may be allowed/invited to select terms him- or herself from any text displayed during the search.

All new terms should be weighted for searching using the relevance weight (or the combined iterative weight, see below). The offer weight is purely for the selection decision.

Iterative combination

The relevance weight may be substituted for the collection frequency weight in the combined weight, formula (2), to give a Combined Iterative Weight:

```
(5)   CIW (i,j)  =  [ RW (i) * TF (i,j) * (K1+1) ] /

                    [ K1 * ( (1-b) + (b * (NDL (j)) ) ) + TF (i,j) ]
```

(Again, if the `CIW` is being implemented, it may also be used for the first iteration instead of `CW`, with `r` and `R` set to zero.)

4 First requests

The entire strategy described so far relies on reasonably good starting requests, or there is no basis for leverage via statistics. Although an invitation to a user to type in a free-form query may result in two or three (or even one) usable terms, it is preferable to start from requests with at least 5 content terms: then at least some will match, and enough will match to e.g. eliminate matches on unwanted word senses, etc.

5 Longer queries

We have hitherto covered only term weighting using information given by individual documents or document sets. It may also be appropriate to use information about the frequency of search terms in requests, as follows.

If you have requests longer than a few words or a sentence, i.e. ones in which query term stems may occur with different frequencies `QF(i)`, then for each query term–document match compute the Query Adjusted Combined Weight:

```
QACW (i)  =  QF(i) * CW(i,j)
```

or Query Adjusted Combined Iterative Weight:

```
QACIW (i)  =  QF(i) * CIW(i,j)
```

Note that when query expansion is used, additional terms all have a query frequency of 1, whatever the frequencies of the original query terms: the Offer Weights used to suggest additional query terms do not reflect within- document frequencies for terms occurring in the retrieved set. On the other hand if searching is restarted using a retrieved relevant document as the new request, this could generate variable query term frequencies.

6 Elaborations

It may be sensible, for some files, to index explicitly on complex or compound terms, e.g. specialised proper names, or fixed multi-word strings, as units, rather than just to hope for conjoint matching at search time. Where a suitable lexicon is available, this can be used to assist in the construction of the inverted file, which will also supply the necessary counting data for weighting. Discovering, by inspection, what multi-word strings there are in a file is a quite different, and very expensive enterprise. It may also be possible to require conjoint matching e.g. for a pair of words, as a preliminary to ranking output, though it may not be practicable to go beyond document co-presence and require text proximity. But it is always important in using multi-word terms to recognise the need to allow for the value of matches on individual components of a compound as well as on the compound as a whole. In general these elaborations are tricky to manage, and are not recommended for beginners.

References

[1] Harman, D.K. (Ed) *The First Text REtrieval Conference (TREC-1)*, NIST SP 500-207, National Institute of Standards and Technology, Gaithersburg, MD, 1993.

[2] Harman, D.K. (Ed) *The Second Text REtrieval Conference (TREC-2)*, NIST SP 500 215, National Institute of Standards and Technology, Gaithersburg, MD, 1994.

[3] Harman, D.K. (Ed) *Overview of The Third Text REtrieval Conference (TREC-3)*, NIST SP 500-225, National Institute of Standards and Technology. Gaithersburg, MD, 1995.

[4] Harman, D.K. (Ed) *The Fourth Text REtrieval Conference (TREC-4)*, NIST SP 500-236, National Institute of Standards and Technology, Gaithersburg, MD, 1996.

[5] Harman, D.K. (Ed) *The Fifth Text REtrieval Conference (TREC-5)*, National Institute of Standards and Technology, Gaithersburg, MD, 1997.

[6] IP&M. Special issue of *Information Processing and Management* on TREC, 31 (3), 1995, 269-429.

[7] Porter, M.F. 'An algorithm for suffix stripping', *Program*, 14, 1980, 130-137.

[8] Robertson, S.E. 'On term selection for query expansion', *Journal of Documentation*, 46, 1990, 359-364.

[9] Robertson, S.E. and Sparck Jones, K. 'Relevance weighting of search terms', *Journal of the American Society for Information Science*, 27, 1976, 129-146; (reprinted in Willett 1988).

[10] Robertson, S.E., Walker, S. and Hancock-Beaulieu, M.M. 'Large test collection experiments on an operational, interactive system', in IP&M, 1995, 345-260.

[11] Robertson, S.E. et al. 'Okapi at TREC-3', in Harman, D. (Ed) *Overview ... (TREC-3)*, NIST SP 500-225, 1995, 109-126.

[12] Willett, P. (Ed) *Document retrieval systems*, London: Taylor Graham, 1988.

See also:
Sparck Jones, K., Walker, S. and Robertson, S.E. "A probabilistic model of information retrieval: development and comparative experiments. Parts 1 and 2", *Information Processing and Management*, 36 (6), 2000, 779-808 and 809-840.