Okapi at TREC–6
Automatic ad hoc, VLC, routing, filtering and QSDR

S. Walker* S.E. Robertson* M. Boughanem* G.J.F. Jones† K. Sparck Jones‡

Advisers: E. Michael Keen (University of Wales, Aberystwyth), Karen Sparck Jones (Cambridge University), Peter Willett (University of Sheffield)

Note on notation

In the tables **P<n>** means precision at cutoff <**n**> documents and **RPrec** means precision after R documents have been retrieved, where R is the number of known relevant documents. *TSV* means Term Selection Value (see Section 3.1).

1 Introduction

Automatic ad hoc

Many experiments were concerned with "blind" expansion (i.e. expansion using pseudo-relevant and -nonrelevant documents). A very large number of runs were done on TREC–3, 4 and 5 data to investigate the effect of varying the Okapi BM25 parameters on precision at low recall. Methods of selecting and ranking topic terms and expansion terms were investigated. In particular, introducing a "non-relevance" component into the expansion term selection function appears to give a small benefit. This work produced good results on TREC-5 data.

Three runs were submitted: long, description, and title only. There was a mistake in the long run. With this corrected, all three runs were among the best, on most statistics.

VLC track

We were interested to find out whether the Okapi BSS (Basic Search System) could handle more than 20 gigabytes of text and 8 million documents without major modification. There was no problem with data structures, but one or two system parameters had to be altered. In the interests of speed and because of limited disk space, indexes without full positional information were used. This meant that it was not possible to use passage-searching. Apart from this, the runs were done in the same way as the ad hoc, but with parameters intended to maximize precision at 20 documents.

Several pairs of runs were done, but only one—based on the full topic statements—was submitted.

Automatic routing

The emphasis was on term weighting using iterative methods on a number of training databases. City's TREC-5 methods were compared with a type of simulated annealing technique. The latter was very greedy, and tended to result in serious over-fitting. Successive runs using the same parameters would result in very different term-weights, and test results bore no predictable relation to training scores. This effect was lessened by merging a number of queries. Eventually it was decided to use a technique based on City's TREC-5 methods followed by a limited amount of annealing, followed by six–twelve-fold merging of queries. Most of the experiments were done using TREC-5 training and test data. It was possible to improve on City's TREC-5 results but only by a few percent.

*Centre for Interactive Systems Research, Department of Information Science, City University, Northampton Square, London EC1V 0HB, UK. email {sw,ser}@is.city.ac.uk, bougha@irit.fr

†University of Exeter. email gareth@dcs.exeter.ac.uk. Currently Visiting Fellow, Toshiba Corporation, Japan

‡University of Cambridge. email Karen.Sparck-Jones@cl.cam.ac.uk

It was difficult to decide what training data to use for the TREC–6 runs. In the event, both the runs submitted only used the TREC–5 FBIS (filtering) data. One run was based on use of the full (filtering) training data both as term source and for deriving weights; for the other, the training set was split into "odd" and "even" with one half used as term source and the other for weighting. The second of these runs was one of the best submitted.

Filtering track

The filtering work was essentially only a small extension of the routing task effort. The pool of merged routing queries was used, but query selection was based on maximizing (over the training data) each of the utility functions for each topic. Two triples of runs were submitted. Both these sets compared very favourably with other participants' results.

QSDR

Some small-scale experiments were run at Cambridge, using Okapi-type methods, with the QSDR data. These tests gave some indication (albeit qualified by the size of the experiment) that the methods are sufficiently robust to give satisfactory performance with appropriate tuning.

2 Okapi at TRECs 1–5

The search systems City have always used for TREC are descendants of the Okapi systems which were developed at the Polytechnic of Central London[1] between 1982 and 1988 under a number of grants from the British Library Research & Development Department and elsewhere. These early Okapi systems were experimental highly-interactive reference retrieval systems of a probabilistic type, some of which featured automatic query expansion [1, 2, 3].

For TREC–1 [4], the low-level search functions were generalized and split off into a separate library — the Okapi Basic Search System (BSS). User interfaces or batch processing scripts access the BSS using a simple command language-like protocol.

City's TREC–1 results were very poor [4], because the classical Robertson/Sparck Jones weighting model [5] which Okapi systems had always used took no account of document length or within-document term frequency.

During TREC–2 and TREC–3 a considerable number of new term weighting and combination functions were tried; a runtime passage determination and searching package was added to the BSS; and methods of selecting good terms for routing queries were developed [7, 8]. During the TREC–2 work "blind" query expansion (feedback using terms from the top few documents retrieved in a pilot search) was tried for the first time in automatic ad hoc experiments, although we didn't use it in the official runs until TREC–3. Our TREC–3 automatic routing and ad hoc results were both relatively good.

TREC–4 [9] did not see any major developments. Routing term selection methods were further improved.

By TREC–5 many participants were using blind expansion in ad hoc, in some cases more successfully than City [10, 11]. In the routing, we tried to optimize term weights after selecting good terms (as did at least one other participant); our routing results were again among the best, as were the filtering track runs.

3 The system

3.1 The Okapi Basic Search System (BSS)

The BSS, which has been used in all City's TREC experiments, is a set-oriented ranked output system designed primarily for probabilistic–type retrieval of textual material using inverted indexes. There is a family of built-in weighting functions as defined below (equation 1) and described more fully in [8, Section 3]. In addition to weighting and ranking facilities it has the usual boolean and quasi-boolean (positional) operations and a number of non-standard set operations. Indexes are of a fairly conventional inverted type. There were again no major changes to the BSS during TREC–6.

Weighting functions

All TREC–6 searches used varieties of the Okapi **BM25** function first used in TREC–3 (equation 1).

[1]Now the University of Westminster.

$$\sum_{T \in \mathcal{Q}} w^{(1)} \frac{(k_1+1)tf}{K+tf} \frac{(k_3+1)qtf}{k_3+qtf} + k_2.|\mathcal{Q}|.\frac{avdl-dl}{avdl+dl} \tag{1}$$

where

$\mathcal{Q}$ is a query, containing terms T
$w^{(1)}$ is either the Robertson/Sparck Jones weight [5] of T in $\mathcal{Q}$

$$\log \frac{(r+0.5)/(R-r+0.5)}{(n-r+0.5)/(N-n-R+r+0.5)} \tag{2}$$

or else a slightly modified and more general version which takes account of non-relevance as well as relevance information [6]

$$\frac{k_5}{k_5+\sqrt{R}}(k_4+\log\frac{N}{N-n}) + \frac{\sqrt{R}}{k_5+\sqrt{R}}\log\frac{r+0.5}{R-r+0.5} - \frac{k_6}{k_6+\sqrt{S}}\log\frac{n}{N-n} - \frac{\sqrt{S}}{k_6+\sqrt{S}}\log\frac{s+0.5}{S-s+0.5} \tag{3}$$

N is the number of items (documents) in the collection
n is the number of documents containing the term
R is the number of documents known to be relevant to a specific topic
r is the number of relevant documents containing the term
S is the number of documents known to be nonrelevant to a specific topic
s is the number of nonrelevant documents containing the term
K is $k_1((1-b)+b.dl/avdl)$
k_1, b, k_2, k_3 and k_4 are parameters which depend on the on the nature of the queries and possibly on the database. For the TREC–6 experiments, k_1 was 1.2 and b was 0.75 except where stated otherwise; k_2 was always zero and k_3 anything from 0 to 1000; when there was not much relevance information -0.7 was a good value for k_4, otherwise zero.
k_5 and k_6 determine, in equation 3, how much weight is given to relevance and non-relevance information respectively. Typical ranges are 0–4 for k_5 and 4–∞ for k_6
tf is the frequency of occurrence of the term within a specific document
qtf is the frequency of the term within the topic from which Q was derived
dl and $avdl$ are the document length and average document length (arbitrary units) resp.

When k_2 is zero, as it was for all the results reported here, equation 1 may be written in the simpler form

$$\sum_{T \in \mathcal{Q}} w^{(1)} \frac{(k_1+1)tf}{K+tf} \frac{(k_3+1)qtf}{k_3+qtf} \tag{4}$$

Nonrelevance information

The extension to the basic weighting formula given by equation 3 is motivated mainly by the desire to make use of explicit judgment of nonrelevance, rather than relying entirely on the "complement" method, by which all documents not known to be relevant are assumed to be nonrelevant. There is a fuller discussion in [6]. This formula might be used in various environments; the particular use reported here has to do with "blind" expansion, where there are no explicit judgments of relevance. Further detail is given below.

Term ranking for selection

In [8] there is a brief discussion of some alternative ways of ranking potential expansion terms. It appeared that no method was superior to the method proposed in [12] by which terms are ranked in decreasing order of $TSV = r.w^{(1)}$. In line with the "nonrelevance" version of $w^{(1)}$ (equation 3) Boughanem has proposed the more general function

$$TSV = (r/R - \alpha s/S).w^{(1)} \tag{5}$$

where $\alpha \in [0,1]$ and r, R, s and S are as above.

Passage determination and searching

Since TREC–3 the BSS has had facilities for search-time identification and weighting of any sub-document consisting of an integral number of consecutive paragraphs. It was described, and some results reported, in [8]. Passage searching almost always increases average precision, by about 2–10 percent, as well as recall and precision at the higher cutoffs. It often, surprisingly, reduces precision at small cutoffs, so is not used in pilot searches for expansion runs. Unless stated, and except for the VLC runs, where the indexes contained insufficient information, and the QSDR runs, passage searching was used in all the runs reported in this paper; comparisons between passage and non-passage runs can be seen in a few of the tables.

3.2 Hardware

There were three dedicated TREC Suns: two 143 MHz single-processor Ultra 1s with an inadequate 128 MB each and an SS20 with 160 MB. Two SS10s were also used, and a third Ultra 1 with 64 MB was borrowed for a time. There was about 60 GB of disk storage, most of it attached to the Ultras and one of the SS10s, and two tape drives. The machines were connected by a local segment of slow ethernet.

3.3 Database and topic processing

For interactive purposes (reported elsewhere) it is necessary to provide for the readable display of documents. Since we have not (yet) implemented a runtime display routine, nor adequate parsing and indexing facilities, for SGML data, all the TREC input text is subjected to batch conversion into a uniform displayable format before further processing. This is done by means of hacked up shell scripts specific to the input dataset. The output records always have three fields: document number, any content unsuitable for indexing (or not to be searched—such as controlled descriptors in some datasets), and the searchable "TEXT" and similar portions.

All the TREC text indexing was of the keyword type. With the exception of the VLC databases a few multiword phrases such as "New York", "friendly fire", "vitamin E" were predefined and there was a pre-indexing facility for the conflation of groups of closely related or synonymous terms like "operations research" and "operational research" or "CIA" and "Central Intelligence Agency". A stemming procedure was applied, modified from [13] and with additional British/American spelling conflation. The stoplist contained 222 words.

Initial topic processing deleted terms such as "document", "describe(s)", "relevan…", "cite…" from any description, narrative or summary fields. What is left was then processed in the same way as text to be indexed. There was a facility for producing adjacent pairs of terms from the topic statements but this was not used during TREC–6. (City have repeatedly experimented with term pairs in previous TRECs, always with negligible benefit.)

4 Automatic ad hoc and VLC track

4.1 "Blind" query expansion

Since TREC–2 City, in common with several other participants, have been experimenting with generating queries for ad hoc searching with the aid of assumed relevant and, more recently, assumed nonrelevant documents, retrieved by means of a pilot search. In general, terms are extracted from the assumed relevant documents and assigned weights using equation 2 or 3. Resulting terms which also occur in the topic statement may have their weights modified by using a positive value of k_3 in the equation, and sometimes all topic terms have their weights increased by a constant factor. Finally, query terms are selected from the resulting pool, usually in descending TSV order. There is of course no need to use the target database alone as term source; it is fairly obvious that, under suitable conditions on the content of the database, the larger the collection the greater will be the density of relevant documents.

We have done a very large number of test runs using blind expansion, but these have resulted in very little in the way of guidelines. In our original TREC–2 experiments (using just the target database for the pilot search) we noted that almost any topic would benefit from *some* expansion, even where it turned out that none of the pseudo-relevant documents was really relevant. Unfortunately, there is very wide variation between topics: an expansion techique which is good for one may be bad for another. We have so far found no effective way of choosing expansion methods to suit individual topics.

There is a quite unmanageable number of independent variables: method of derivation of the pilot query, pseudo-relevance and -nonrelevance decisions, extraction and weighting of terms, etc. We choose the parameters of the pilot search (k_1, b, k_3, k_4) to maximize (we hope) precision at cutoff R, where R is the intended number of pseudo-relevant documents. We then assume that documents 1–R are relevant, skip the next G and assume the following S are non-relevant (having retrieved $R + G + S$ documents). Sometimes, long documents, over 10,000 bytes say, are removed from the R-set[2]. Terms are extracted from each remaining document[3] in the R-set, and n, r and s determined (see Section 3.1). These term records are then run against the topic statement and records for topic terms have their within-topic frequency added. We now have a set of terms, perhaps numbering several thousand, for each topic, from which queries can be constructed.

Many different queries can be constructed from a single term-set. All six[4] of the BM25 parameters (Section 3.1) can be varied. The weight bonus percentage T given to topic terms can be varied, as can the method of term selection. We have tried the following selection methods. In all cases terms are weighted and arranged in descending TSV order.

1. The top t terms are selected.
2. All terms with TSV above a fixed threshold, or above a fixed proportion of the greatest TSV value, are selected.
3. All topic terms are selected together with the best a non-topic terms.
4. If there are nt topic terms, all topic terms are selected together with the best A% of nt non-topic terms.

Refinements may be added to any of 1–4 above, either to improve results or for efficiency reasons. These include

- exclude terms with very large n
- exclude non-topic terms with small n
- exclude terms with low r
- exclude non-topic terms containing a digit.

4.2 Ad hoc runs

These are summarised in Table 1. All reported runs used passage searching, which gave gains of up to about 12% in average precision. All runs also excuded non-topic terms containing a digit, and in some cases there were rather weak restrictions on r and n.

Three official runs were submitted: city6al (long), city6ad (short) and city6at (title). All used blind expansion from initial searches of a database made up of disks 1–5 and the TREC–4 routing data. The procedures used were based on those which had been most successful in trials with TREC–2, 3 and 5 data. We report also runs using title + description fields.

- The long run used the top 30 terms from the top 15 documents retrieved in the pilot search, documents longer than 10,000 characters being discarded. The terms were weighted using equation 3 with $G = S = 500$, $k_5 = 1$ and $k_6 = 64$. Topic terms had their weights multiplied by 2.5. The terms were ranked by TSV with $\alpha = 0.15$ in equation 5. There was a mistake in the pilot run script, k_3 being treated as zero instead of 7 as intended, so versions of this run have been repeated with two corrected termsets[5].
- The title + description run used the top 30 terms from the top 10 documents retrieved in a pilot search of the disks 1–5 database with pilot $k_3 = 7$ and documents longer than 10,000 characters being discarded. The terms were weighted using equation 3 with $G = S = 500$, $k_5 = 1$ and $k_6 = 64$. $k3 = 1000$ in the final search.
- The official short (description) run used the top 24 terms from the top 10 documents retrieved using $k3 = 7$ and k_4=0, documents longer than 10,000 characters being discarded. The terms were weighted using equation 3 with $G = S = 500$, $k_5 = 1$ and $k_6 = 128$. Topic terms had their weights multiplied by 2.5. The alternative expansion method reported in the table used $k5 = 2$, $k_6 = 64$ on topic (description) terms + an additional 1.75 times the topic length of non-topic terms.
- The title run used topic terms + the top 20 non-topic terms from the top 7 documents retrieved using $k_3 = k_4 = 0$, documents longer than 10,000 characters being discarded. The terms were weighted using equation 3 with $G = S = 500$, $k_5 = 1$ and $k_6 = 128$. Topic terms had their weights multiplied by 3.5.

[2]This does not seem to be beneficial, but does speed the process of query construction.
[3]Sometimes terms are extracted just from the "best" passage in each document, but again we have not found this of noticeable benefit.
[4]Or seven, but k_2 is nearly always zero.
[5]The corrected termsets were derived from a slightly smaller database consisting of disks 1, 2, 3, 4 and 5.

Table 1: Automatic ad hoc results

Method	AveP	P10	P15	P20	RPrec	Rcl
Long topics						
Official city6al, described in 4.2	0.233	0.394	0.357	0.332	0.260	0.525
Corrected version of city6al (pilot $k_3 = 7$)	0.262	0.480	0.435	0.393	0.286	0.581
As previous + final $k_3 = 1000$	0.298	0.508	0.459	0.435	0.310	0.595
As previous but pilot $k_3 = 1000$ and pilot $k_4 = -.7$	0.305	0.522	0.467	0.430	0.322	0.602
No expansion, $k_3 = 1000$, $k_4 = -.7$	0.288	0.480	0.436	0.402	0.320	0.581
Title + description						
Described in 4.2	0.298	0.484	0.443	0.399	0.324	0.582
No expansion, $k_3 = 1000$, $k_4 = -.7$	0.282	0.450	0.405	0.370	0.318	0.544
Description only						
Official city6ad, described in 4.2	0.216	0.356	0.309	0.283	0.250	0.426
Alternative expansion method	0.226	0.354	0.315	0.293	0.257	0.441
No expansion, $k_3 = 1000$, $k_4 = -.7$	0.210	0.320	0.299	0.281	0.249	0.440
Short topics (title only)						
Official city6at, described in 4.2	0.288	0.438	0.393	0.367	0.322	0.555
No expansion, $k_3 = k_4 = 0$	0.251	0.408	0.355	0.336	0.296	0.502

4.3 Very large collection (VLC) track

This was quite a relaxation after the other tracks. There were no preliminary experiments. Any problems were logistical. The Okapi system had never been tried on a database more than a quarter of the size of the new compendium. Two modifications had to be made. Since Sun operating systems (prior to Solaris 2.6) do not allow files to exceed 2 GB Okapi database textfiles may comprise a number of physically separate volumes, the maximum number of volumes had to be increased from 8 to 16 to cater for the 20 GB of text. Secondly, it was clear we would not have enough free disk space, at least not locally, to store the text and an estimated 12 GB of inverted indexes. Hence it was decided to create a new form of index which would not contain full within-document positional information. This brought the index overhead down from about 60% to about 20% of the textfile size. Once these alterations had been done there only remained the operational complications of scheduling the tape-reading, decompression, conversion via Okapi exchange format to runtime format; followed by parsing and term generation (done in five or six portions on several machines), and finally merging of the resulting streamers and the production of a dictionary and single inverted file (in three volumes).

VLC results

These are summarised in Table 2. Note that except for the official runs a considerable number of non-assessed documents were retrieved, so it is likely that "true" results are somewhat better than the ones shown.

The official run, city6vl, (and the baseline run city6vbl), used the top 25 terms by *TSV* (method 1 in Section 4.1) from the top 15 documents retrieved by a pilot search using the full topic statements with $k_4 = 0$, documents longer than 10,000 characters being discarded. Terms were weighted using equation 3 with $k_5 = 1$ and $G = S = 0$, and topic terms had their weights multiplied by 2.5. Final k_3 was zero. There were two mistakes in the scripts which executed this run: the pilot k_3 should have been 7 but was treated as zero, and there was a fault in the term ordering procedure. It was not possible to use passage searching because of the absence of positional information in the indexes. This run came second on precision at cutoff 20 in the official results (ignoring an impressive but presumably manual run from the University of Waterloo).

When the mistakes were discovered a new termset was generated, and gave the results shown under "Corrected city6vl". This gives worse precision at 20, although better on the other statistics. A run using reweighted topic terms only was impressive, and a run with no expansion at all did beter than most of the expanded ones. The best result of all was obtained by using a final k_3 of 1000 and the method of the official city6vl.

In the baseline test, the official run was poor, but a run with no expansion was surprisingly good.

Table 2: VLC results

Method	P5	P10	P15	P20	% unassessed docs
Official city6vl	0.584	0.562	0.532	0.515	0.0
Corrected city6vl	0.592	0.578	0.547	0.506	19.1
As city6vl with final $k_3 = 1000$	0.668	0.614	0.585	0.548	8.8
As corrected city6vl with final $k_3 = 1000$	0.612	0.594	0.564	0.526	14.8
As corrected city6vl, final $k_3 = 1000$, reweighted topic terms only	0.648	0.600	0.556	0.529	14.3
No expansion, $k_3 = 1000$, $k_4 = -.7$	0.672	0.618	0.565	0.517	13.6
As previous but $n < 800000$	0.656	0.582	0.551	0.517	14.4
Official baseline city6vbl	0.404	0.382	0.337	0.320	0.0
Baseline no expansion, $k_3 = 1000$, $k_4 = -.7$	0.520	0.440	0.407	0.374	9.8

VLC timings

The figures in Table 3 are approximate wall-clock times, corrected where necessary to as near as possible what they would have been if all run on one of our Ultra 1s. No correction has been attempted for varying network, disk and CPU loadings from unassociated processes. As expected, most times look roughly linear in data size (the baseline data consisted of a systematically sampled 10% of the main task data). There should be a rather small logarithmic component in the indexing times: this would doubtless show up in the CPU times. It is not altogether clear why expansion term generation took relatively longer for the baseline task, but the explanation may lie in the very inefficient scripts used which have an overhead proportional to the number of topics.

Table 3: VLC processing times in wall-clock hours, main and baseline tasks

	Time (hours)	
Process	main	baseline
Uncompress raw data	3.3	0.3
Strip and convert uncompressed data to Okapi runtime format	28.9	2.4
Parse and index	70.8	7.2
Generate expansion term pool and queries	17.0	2.6
Execute 50 25–term queries	1.0	0.1

4.4 Discussion of ad hoc

City's description and title runs did well, and the corrected long topic runs also compare very favourably with other TREC–6 results. It is quite clear that "blind" query modification is beneficial provided that a large enough database is available, even when use of the pseudo-relevant documents is limited to reweighting the original query terms. However, most of the more successful participants are using something similar. Passage searching almost always increases average precision and recall, and the new weighting formula equation 3 is undoubtedly of some benefit [6], although this is not demonstrated in this paper. The new formula has two advantages: smallish non-zero values of k_5 mean that limited use can be made of little, or dubious, positive relevance information; and it appears that quite large k_6 values enable some use to be made of negative information.

5 Automatic routing and filtering

5.1 Routing

Training sets

To start with, all the (positive) relevance judgments for the TREC–6 routing topics were used. It is probably inadvisable to use datasets with incomplete relevance judgments for routing term selection, so about six databases had to be used, involving a lot of time, both human and machine. For most topics there were a large number of relevant documents (mean 579, median 510). After a number of trials had been done using the full set of judgments it was decided to try using the filtering training set (TREC-5 routing database) only, although the judgments were said to be incomplete for nine of the 47 topics. For this set the mean was 123 (median 45).

Outline

Terms were extracted from relevant documents for some database and weighted using the customary Okapi formula $w^{(1)}$ (equation 2; the new "nonrelevance" formula equation 3 was not used in the routing). The terms were then arranged in descending TSV order. A fixed number of terms were taken from the top of the term list and subjected to selection and/or reweighting procedures. Sets were formed and scored in some way using either the same or a different database and set of judgments with the object of obtaining as high a score as possible. The only scoring function used for TREC–6 was non-interpolated average precision at cutoff 1000. Finally, a number of the resulting query sets were merged.

Term weight optimization

For TREC–5 we tried some reweighting experiments, which appeared to give a small improvement over simple selection of terms. Since we had acquired two extra, and faster, computers this year it was decided to try some more drastic reweighting methods. It seemed possible that some type of simulated annealing might work, although it might be thought that practical scoring functions would not have the relative "smoothness" for this to work satisfactorily. A simple but very compute-intensive procedure was developed and gave encouraging scores during reweighting. However, when applied predictively results varied rather wildly. It appeared that the procedure was giving serious overfitting to the selection database. The next step was to combine the new procedure with the deterministic reweighting which we had used for TREC-5. Eventually we tried two to four passes of deterministic single-term reweighting followed by a rather mild annealing process. The latter almost always gave a noticeable increase in selection score but it is not yet clear whether there was any real gain when applied predictively to the test set.

The simulated annealing procedure

This proceeds in a number of stages, the "temperature" being reduced between each one, with a final "quench" at zero temperature. At all times two configurations are saved: a local "best" and a global best. Each stage consists of a number of iterations in each of which the weights of randomly selected terms are increased or decreased and a new score calculated. If the new score is higher than the local best the new configuration is retained and becomes the current best. If it is lower than the current best it is nevertheless retained (as current best) with a probability which decreases with T, the current temperature[6]. The motivation for sometimes keeping an apparently worse configuration is that this may enable escape from a local maximum. The hope is that a lot of local maxima may be explored and one ends up with the best one, or at least a rather good one. Finally, if the local best from the "quench" stage is worse than the global best the latter becomes the final result.

Obviously, there are a considerable number of tuning parameters: for example the distributions of the number of terms to have their weights altered and the extent of weight variation, the temperature reduction function, rules for ending stages and for ending the whole procedure. We were not able to explore the possibilities anywhere near exhaustively, and results so far are not particularly encouraging.

Merging runs

As in previous TRECs, a number of term sets from different selection procedures were merged to form the final query, thus reducing over-fitting. Where partitioned databases were used for training all runs were duplicated: terms from "odd" (say) and optimization on "even" followed by the reverse; the two term sets would then be merged. The simulated annealing led to quite wild variation in the magnitude and range of weights, so a term set to be merged with another would have its weights normalised relative to the median weight of the set.

5.2 Automatic routing results

These are given in Table 4.

[6]The usual rule, which we used, is "accept worse score with probability $\exp(-(\mathit{best\_score} - \mathit{new\_score})/T)$".

As mentioned above, both sets of submitted queries (city6r1 and city6r2) were derived using only the TREC–5 routing database and the TREC–6 filtering training relevance judgments. Had time allowed we should probably have used some additional pre–TREC–5 training information for some of the topics with few known relevant FBIS documents. The difference between the two sets is that for city6r1 the terms came from one half of the database and the optimization was done on the other half; whereas for city6r2 the whole database was used both as term source and for optimization. Experiments with TREC–5 routing data had suggested that the former method was likely to give slightly better results, although the relatively small number of TREC–6 training judgments made it dangerous to assume that this would still hold. Hence it is quite surprising that the cityr1 result turned out so much the better of the two. However, there were other differences between them. Both sets of queries were formed by merging a number of query sets, but 24 (12 pairs) were used to form city6r1 and only six for city6r2; city6r1 used four deterministic weight variation passes with just a final "quench" stage; city6r2 had three deterministic passes followed by four stages of simulated annealing.

All the optimization runs started with a term pool of size 100 (in previous TRECs we had found a small gain from using up to 300 terms, but the simulated annealing would have been too slow on sets of this size). City6r1 ended with a mean of 138 terms per query and city6r2 with 86, but many terms had very low weights and the queries could probably be reduced by 25 percent or more without greatly affecting results.

Table 4: Automatic routing results

Run	AveP	P5	P10	P15	P20	P30	RPrec	Rcl
city6r1	0.408	0.698	0.653	0.606	0.578	0.548	0.411	0.809
As city6r1 but without passages	0.399	0.706	0.651	0.604	0.581	0.545	0.409	0.802
city6r2	0.378	0.668	0.626	0.590	0.554	0.523	0.399	0.760
As city6r2 but without passages	0.368	0.672	0.619	0.580	0.555	0.515	0.392	0.753

5.3 Filtering

Filtering estimation procedure

A pool of queries was selected from the merged queries produced during the routing training. Each query was executed against a training database to produce a standard TREC output file with the addition of a field for each document containing the relevance assessment (relevant, nonrelevant or not assessed). These output files were run through a script which calculated the value of each of the utility functions at each rank. The threshold weight which maximized each function was then found. Output from this stage was of the form

$$\langle\text{topic}\rangle\langle\text{func}\rangle\langle\text{value}\rangle\langle\text{thresh}\rangle\langle\text{rank}\rangle\langle\text{prec}\rangle\langle\text{recall}\rangle\langle\text{\# rels}\rangle\langle\text{query file}\rangle$$

The lines which gave the highest value for each function for each topic were extracted from this, thus giving the queries and thresholds to be used.

Filtering results

Three triples of queries and thresholds were submitted, city6f1[1-3] and city6f2[1-3]. City6f1 had what ought to have been a sounder basis; it was produced by executing routing queries—whose weights had been derived using one half of the training database—against the other half of the database. City6f2 used queries where both halves of the database had contributed to the weights, run retrospectively against the whole database. The latter must have led to a substantial overestimation of maximum function values, though it is not obvious what effect this would have on the estimation of thresholds. From the comparisons with median scores (Table 5) it looks as if city6f2 may in fact have been slightly the better of the two.

5.4 Discussion of routing results

One of the most important aspects is avoidance of over-fitting of queries to the training database. Given enough computer time it is not difficult to obtain very good retrospective scores, but the resulting queries produce poor results when applied predictively. We have tried various appealing ideas along the lines of rejecting rare terms or only taking "good" passages from long relevant documents, but have always found these to be if anything marginally

Table 5: Filtering results (pooled evaluation): comparison with median scores

Run	function	$\geq$ med.	best	$<$ med.	worst
city6f11	F1	36	8	11	0
city6f21	F1	37	4	10	0
city6f12	F2	37	5	10	0
city6f22	F2	41	5	6	0
city6f13	ASP	45	5	2	0
city6f23	ASP	45	8	2	0

detrimental. In practice, it seems to be best to merge queries from as many reasonably promising sets as possible. It may be that over-fitting was less of a problem in TREC–6, where for the first time the test database was (presumably) rather "similar" to the (filtering) training database, a more realistic setup than in previous TREC routing tracks.

From the point of view of real life routing situations, there is an urgent need to work on techniques for properly using relevance information as it increases from an initial zero. Our results are not good on topics for which there are few relevant documents. Blind expansion is no good unless there is a large database of documents of a suitable type. In particular some experiments should be done on query optimization with little relevance information. During our TREC–5 ad hoc work we did some runs where queries were "optimized" on pseudo-relevant documents, with surprisingly good results.

6 QSDR experiments for City University

We were unable at Cambridge, for independent reasons, to do the actual SDR speech test. We therefore carried out only some rather simple, lightweight tests using the baseline SRT data. But these were of the kind drawing on IR experience that the QSDR option was intended to encourage. Thus we treated QSDR as a way of checking out the Okapi-type retrieval methods, already applied to speech data in the Cambridge VMR project [19], with (a) new, different, document data (b) known-item searching rather than conventional queries. We set aside questions of whether the Cambridge speech recognition system could do any better than the one used for the baseline (without or with tuning to the application), and also any retrieval strategies tied to speech (eg various fusion ones combining different recognition strategies).

Thus for both the baseline SRT and the LTT versions of the documents we compared search performance for unweighted terms (UW), terms with only collection frequency weighting (CFW), and terms with full Okapi-style combined weighting (CW, aka BM25, see equation 1 or [20]). We were interested in whether relative performance for these strategies was the same for the SRT and LTT data, and how the weighting formulae in particular behaved in (a) improving targeting on the known item and (b) compensating for speech recognition deficiencies.

However the small data scale, and especially tiny training query sample, made adaptation to the TREC case uncertain, and the results given below must be taken with much salt. The small document set size also limits inference for performance with the larger test query set. Using the training data we applied, as usual, stop listing (with the standard van Rijsbergen stop list) and stemming (Porter), and after experiment set the CW tuning constants b and K. Some method for dealing with acronyms is needed, but without any in the training query set we were not able to choose a suitable one.

Performance for the training set (though only indicative, with so few queries) suggested that respectable performance for SRT against LTT, and decent performance for the known-item type searching, should be obtained for the test data, as well as the predicted best results for CW. This was borne out in practice, as shown below. In particular, there are clear performance gains for CW with the SRT test data, and though the expected run length is still much worse for SRT than LTT, CW again gives the best results.

Table 6 illustrates CW performance with parameter settings the same for the LTT and SRT data. The test data runs labelled cw2, with the parameter settings based on the training data, are the officially submitted runs 'citysdrR2' and 'citysdrB2', for LTT and SRT respectively. Setting the parameters differently for LTT and SRT, as in the runs citysdrR1 and citysdrB1, makes little difference, as predictable for such small data. On the other hand, as the cw4 figures in Table 6 show, setting the CW parameters to suit the test data rather than applying training data ones as in the official runs, could be expected to improve performance; and of course such document-file linking for weighting formulae for ad hoc retrieval is wholly feasible.

Performance for the official runs, taking citysdrR2 and city sdrB2 as representative and using percent queries retrieving the known item at rank 1, was above the median, in the leading cohort with roughly similar performance.

This was reassuring. However, while our tests could be viewed as checking the relative values of weighting formulae, and consistency under scaling up, what we were really testing was the viability of a completely routine approach to spoken document retrieval without any adaptation, in either speech processing or retrieval mechanisms, to the spoken document data and the particular retrieval task. Thus IBM's recogniser was deliberately not tuned to the data, but run blind, to produce the SRT transcripts; and we did not attempt to tune the retrieval system to known-item searching, e.g. by adopting a precision-oriented strategy. Our results therefore suggest that given a good recogniser, like IBM's, and sound retrieval methods, respectable performance can be obtained, though it is possible that additional strategies e.g. data fusion on the speech side or query expansion on the retrieval side, could be of use. However the inference just drawn about competitive performance must be heavily qualified because the retrieval test was so tiny, and the task defined by the requirement to retrieve the specified known items seems to have been rather easy.

Table 6: QSDR results

		Training data			Test data			
		uw	cfw	cw2[a]	uw	cfw	cw2[a]	cw4[a]
Ave Prec	LTT	0.70	0.69	0.78	0.55	0.74	0.81	0.84
	SRT	0.57	0.65	0.65	0.46	0.63	0.68	0.72
Exp Run Lngth	LTT	5.67	7.33	1.50	13.02	5.98	5.41	5.06
	SRT	12.17	9.17	5.00	33.27	16.51	14.29	12.84
Mean Recip	LTT	0.70	0.69	0.81	0.55	0.74	0.81	0.84
	SRT	0.51	0.68	0.73	0.44	0.61	0.68	0.72

[a]setting constants b and K the same for LTT and SRT
cw2 for training data: $b = 0.25$, $K = 2.5$
cw2 for test data uses the training settings
cw4 uses settings chosen for the test data: $b = 0.5$, $K = 1.0$

References

[1] Mitev, N.N., Venner, G.M. and Walker, S. *Designing an online public access catalogue: Okapi, a catalogue on a local area network.* British Library, 1985. (Library and Information Research Report 39.)

[2] Walker, S. and Jones, R.M. *Improving subject retrieval in online catalogues: 1. Stemming, automatic spelling correction and cross-reference tables.* British Library, 1987. (British Library Research Paper 24.)

[3] Walker,S. and De Vere, R. *Improving subject retrieval in online catalogues: 2. Relevance feedback and query expansion.* British Library, 1990. (British Library Research Paper 72.) ISBN 0-7123-3219-7

[4] Robertson, S.E. *et al.* Okapi at TREC. In: [15], p21–30.

[5] Robertson, S.E. and Sparck Jones K. Relevance weighting of search terms. *Journal of the American Society for Information Science* 27, May–June 1976, p129–146.

[6] Robertson, S.E. and Walker S. On relevance weights with little relevance information. In *Proceedings of the 20th annual international ACM SIGIR conference on research and development in information retrieval.* Edited by Nicholas J Belkin, A Desai Narasimhalu and Peter Willett. ACM Press, 1997. p16–24.

[7] Robertson, S.E. *et al.* Okapi at TREC–2. In: [16], p21–34.

[8] Robertson, S.E. *et al.* Okapi at TREC–3. In: [17], p109–126.

[9] Robertson, S.E. *et al.* Okapi at TREC–4. In: [18], p73–96.

[10] Beaulieu, M.M. *et al.* Okapi at TREC–5. In: [11].

[11] *The Fifth Text REtrieval Conference (TREC–5).* Edited by D.K. Harman. Gaithersburg, MD: NIST, 1997.

[12] Robertson, S.E. On term selection for query expansion. *Journal of Documentation* 46, Dec 1990, p359–364.

[13] Porter, M.F. An algorithm for suffix stripping. *Program* 14 (3), Jul 1980, p130-137.

[14] Robertson, S.E., Walker, S. and Hancock-Beaulieu, M.M. Large test collection experiments on an operational, interactive system: OKAPI at TREC. *Information Processing & Management* 31 (3), p345–360, 1995.

[15] *The First Text REtrieval Conference (TREC–1).* Edited by D.K. Harman. Gaithersburg, MD: NIST, 1993.

[16] *The Second Text REtrieval Conference (TREC–2).* Edited by D.K. Harman. Gaithersburg, MD: NIST, 1994.

[17] *Overview of the Third Text REtrieval Conference (TREC–3).* Edited by D.K. Harman. Gaithersburg, MD: NIST, 1995.

[18] *The Fourth Text REtrieval Conference (TREC–4).* Edited by D.K. Harman. Gaithersburg, MD: NIST, 1996.

[19] Jones, G.J.F., Foote, J.T, Sparck Jones, K. and Young, S.J. *Video Mail Retrieval using voice: Report on topic spotting*, Technical Report 430, Computer Laboratory, University of Cambridge, 1997.

[20] Robertson, S.E. and Sparck Jones, K. *Simple, proven approaches to text retrieval*, Technical Report 356, Computer Laboratory, University of Cambridge, updated May 1997.