

C-TEST: Supporting Novelty and Diversity in Testfiles for Search Tuning

David Hawking
Funnelback Pty Ltd
Canberra, Australia
david.hawking@acm.org

Tom Rowlands
CSIRO ICT Centre
and ANU Dept. Comp. Sci.
tom.rowlands@csiro.au

Paul Thomas
CSIRO ICT Centre
paul.thomas@csiro.au

ABSTRACT

Tuning a search facility such as a Web search engine, or an enterprise search tool deployed in a particular organisation, is an economically important activity. Intuitively, an important end goal of tuning should be to maximise satisfaction across the searchers who will use the facility. Tuning should therefore use an unbiased sample of actual search requests, a judging process accurately modelling that of real searchers, and measures optimally correlated with real satisfaction. The process must reflect that the result set for a particular query may be judged by submitters with different underlying information needs. It should also model users' impatience with duplicate results. Here, we describe a formally-defined, practical, public-domain testfile format suitable for use in batch tuning but capable of reflecting these important aspects. We hope that the formats will soon be supported by all major academic IR systems. DTDs for test and result files, plus an associated toolkit and C-TEST example testfiles for some TREC tasks, are available at es.csiro.au/C-TEST.

Keywords

Enterprise search; Search evaluation; Information retrieval

1. INTRODUCTION

Tuning for search result quality is commercially important in both Web and enterprise search. On the Web, higher quality results bring more traffic and more advertising revenue. In the enterprise domain, organisations require effective retrieval to disseminate information, assist potential customers to find products, ensure customers are not lost to competitors through failed search, and to increase employee productivity. Such a system can expect to see a wide range of users—employees, customers and, perhaps most importantly, potential customers—with a wide variety of needs.

An enterprise search tool may ship with default parameters set by tuning using testfiles representing a diverse range of organisations.

In a tuning exercise for a modern search engine there are so many possible combinations of parameter values that it is not possible to tune using interactive evaluations with human subjects. We must capture the essence of evaluation by humans in a testfile capable of being run thousands of times in batch mode.

Testfiles should therefore use an unbiased sample of actual search requests, a judging process accurately modelling that of real searchers, and measures optimally correlated with real satisfaction.

Testfiles should model all the plausibly supported search scenarios—intelligence gathering or academic research, topic distillation, homepage finding, known-item retrieval, aspect recall and so on. These scenarios differ in judging depth, number of answers sought, and the spread of utility values associated with different answers.

Testfiles should also allow for the possibility that some queries are more important than others and that the same query may arise from distinct information needs.

1.1 Modelling satisfaction / dissatisfaction

Intuitively, the end goal of tuning should be to maximise satisfaction generated by the facility when it is deployed. Sometimes in an enterprise search setting the goal is to maximise the satisfaction of the publisher rather than the consumer of information. For example, a publisher may wish to promote items with higher profit margins or greater stock. However, for simplicity, let us assume that publisher satisfaction coincides with overall searcher satisfaction.

1.1.1 Averaging across searchers

If the j -th searcher's satisfaction with the facility F can be represented as s_j on a ratio scale [15] covering the range 0–1, then overall satisfaction with the facility S_F may be computed as an average of the satisfaction scores s_j of each person, $p \in P$, submitting queries.

$$S'_F = \frac{\sum_{j=1}^{|P|} s_j}{|P|}$$

Unfortunately, there are problems with this. First, we have no realistic way of measuring human satisfaction on a ratio scale. We must settle for surrogate measures such as NDCG [8], derived from a very limited ordinal scale used by human judges: e.g. *very useful*, *useful*, *not useful*. We can offer no solution for this defect, and unfortunately, it is shared by all batch evaluation systems now extant.

Second, a person's satisfaction may not be a simple average of their per-query satisfactions; failed queries (for example,

those resulting in a satisfaction score of zero) may weigh heavily in overall satisfaction. This may be particularly true if the failed query arose from a very important information need.

Third, sampling processes may leave us with an incomplete picture of a user’s interactions with the search system.

1.1.2 Averaging across query instances

An alternative approach to computing a facility satisfaction score is to compute a simple average over query submissions, where n is the number of query submissions:

$$S_F = \frac{\sum_{i=1}^n s_i}{n}$$

However, optimising this average will not necessarily eliminate *failures*, where a failure is defined as $s_i < \epsilon$ or, most simply, as $s_i = 0$. As noted above, failures give a particularly bad impression of the search facility, and should perhaps be dealt with separately. In 5.2 we propose that a search facility be optimised first for minimum failure rate and second for maximum average satisfaction.

If there are no users present to tell us how satisfied they were by responses to the queries being studied, e.g. we are evaluating queries taken from a log, then we must take into account that a query may have multiple interpretations. In such a case, it may be desirable to ensure that the facility doesn’t fail for *any* of the interpretations. This scenario is further discussed in Section 5.2.

1.1.3 Contribution

In the present work, we describe a number of major challenges in supporting meaningful search engine tuning. We also present the initial version of a practical, public-domain testfile format (C-TEST) suitable for batch tuning, while at the same time supporting to a considerable degree the various desiderata canvassed above. The C-TEST formats and tools are described in Section 4 while C-TEST support for redundancy and interpretations are discussed in Sections 5.1 and 5.2.

2. RELATED WORK

In this section, we outline previous work both on testfile formats and novelty and diversity.

2.1 Testfiles

The TREC¹ ad-hoc retrieval task is in many ways the archetypal information retrieval evaluation. Its resources have provided data for tuning many information retrieval systems.

In each TREC campaign a set of 50 structured topics were distributed to participants. Topics were invented by humans engaged by the National Institute of Standards and Technology (NIST). An SGML-inspired notation was used to identify title, description, and narrative for the topics. (See Figure 1 for an example.) Participants sometimes used the whole topic, or fields of the topic directly as a query to their retrieval system. In other cases, the topic was considered to be a statement of the person’s retrieval need and a manual or automatic process was used to derive a query from it. After runs were submitted and judged, a *qrels* file, a simple textfile,

```
<top>
<num> Number: 151
<title> Topic: Coping with overcrowded prisons
<desc> Description:
The document will provide information on jail and
prison overcrowding and how inmates are forced to
cope with those conditions; or it will reveal plans
to relieve the overcrowded condition.
<narr> Narrative:
A relevant document will describe scenes of overcrowding
that have become all too common in jails and prisons
around the country. The document will identify how
inmates are forced to cope with those overcrowded
conditions, and/or what the Correctional System is
doing, or planning to do, to alleviate the crowded
condition.
</top>
```

Figure 1: TREC Ad Hoc Topic number 151

```
...
151 0 AP880510-0247 0
151 0 AP880512-0192 0
151 0 AP880514-0176 0
151 0 AP880515-0009 1
151 0 AP880516-0036 0
151 0 AP880516-0081 0
151 0 AP880516-0128 0
...
```

Figure 2: A section of the qrels file for TREC Ad Hoc Topic number 151

¹trec.nist.gov

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<topics>
  <topic lang="fr">
    <identifiant>10.2452/451-AH</identifiant>
    <title>L'arm'ee romaine en Grande-Bretagne</title>
    <description>Trouver des livres ou des publications
      sur l'invasion_et_l'occupation de la Grande-
      Bretagne par les Romains.</description>
  </topic>
  ....
</topics>

```

Figure 3: Example topic from CLEF-2008. Note: well-formed XML, encoding specified, topic language specified

was circulated. (See Figure 2 for an example.) The qrels file contains one line for every human judgment made. Each line consists of topic-id, an unused field, document-id and judgment, separated by whitespace.

In some TREC Web Track campaigns, topics were reverse engineered from queries selected from logs.

Many participants used the topics and qrels from previous ad hoc and Web Track campaigns to tune their systems. However:

- The TREC topics are not an unbiased sample of any real population of users. Substantial performance differences were observed from one year to the next.
- Document judgements are independent of each other. No information is given to reward diversity, novelty or the avoidance of duplicates.
- Topics generally specify a single interpretation of an ambiguous (Web Track) query. A retrieval system is given no credit for responding to a different interpretation.
- The topics and qrels format does not specify the depth of results to be judged.
- As seen in Figure 1 parsing of topics is not as simple as it might be due to unclosed tags and the presence of extraneous words, such as ‘Narrative:’ within elements.

2.1.1 Other evaluation campaigns

Other well-known evaluation campaigns have tended to follow the TREC lead. In CLEF, NTCIR and FIRE, topics and qrels are stored in separate files and, in each case, the qrels file retains the four-column TREC format. Topic files in all these cases conform more closely to normal XML usage, but it is only recently (CLEF-2008 and FIRE) that topic files have included an XML header specifying the character encoding. See Figure 3 for a CLEF example.

Naturally, INEX leads the way in XML conformance of topics; see Figure 4. Interestingly, though, INEX retains a non-XML format very similar to TREC qrels for judgments, shown in Figure 5.

2.2 Novelty and diversity

The probabilistic model introduced by Maron and Kuhns [11] was criticised by Cooper (reported by Robertson [12]) for only considering the dominant class of users for a given request. Were the same request issued by classes of users with contradictory views of utility, the resultant document ranking

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd">
<inex_topic topic_id="405" ct_no="197">
  <title>"The_Old_Man_and_the_Sea"</title>
  <castitle>/**[about(.//collectionlink,"the_old_man_and_
    the_sea")]</castitle>
  <description>Looking for references to the book "The_
    Old_Man_and_the_Sea"</description>
  <narrative>I am writing an essay on the book "The_Old_
    Man_and_the_Sea" and would
    like to get all the contexts in which the book is
    referenced.</narrative>
  <ontopic_keywords>Old Man Sea Hemingway Ernest</
    ontopic_keywords>
</inex_topic>

```

Figure 4: Example topic from INEX. Note: well-formed XML, encoding specified, use of a DTD, use of element attributes, and title in extended NEXI

```

544 Q0 20347 26294 26299 1 1:26294
544 Q0 261763 0 2258

```

Figure 5: Example INEX judgement file

would always side with the dominant class. The probabilistic model was not considering diversity in the viewpoint of the user.

Diversity can be found not only in users and their interpretation of queries, but in sources, languages, time, location and elsewhere. Encouraging a diversity in sources on an enterprise search system has been addressed by applying a decay function to the document scores of sites that already appear.² Google acknowledges the importance of site diversity by presenting pages from the one site in a semi-hierarchical fashion, offering further results from the site only if asked. Del Corso et al. describe a news ranking system that decays the importance of articles as a function of time and also considers each article’s source [6].

A complementary problem to promoting diversity is the reduction of redundancy. The TREC Novelty track [7] posed the task of exposing relevant, non-redundant, novel sentences, given a set of sentence-separated documents. Bernstein and Zobel determined there is a significant number of duplicate documents in the TREC GOV and GOV2 collections [1]. These duplicates, were they to appear twice in a result set considered by an evaluation measure, would skew the evaluation.

Redundancy in information is not necessarily bad. Finding the same information in two places may increase some users’ confidence [4]. Magnini et. al [10] used redundancy in Web documents to build an answer validation system for the TREC question answering track [18].

Clarke et al. proposed a modification of NDCG for document evaluation linked by ‘information nuggets’ [5]. Their evaluation measure further incorporated a parameter accounting for the chance of judge error. They demonstrated the new measure on the TREC Question Answering track data.

Sanderson documents the construction of a test collection designed to demonstrate the ability of retrieval systems to

²Personal communication

deal with ambiguous queries [14].

3. DESIRABLE PROPERTIES OF TESTFILES FOR TUNING

C-TEST was designed with the following desiderata in mind. Testfiles should:

- be represented in unambiguous, easily parsable and human understandable format (XML [2])
- specify the character set used, as the retrieval system needs to know
- allow a heterogeneous mix of search scenarios
- specify all the important parameters of each evaluation
- support both the case where answers are known in advance and where they are not
- allow for multiple interpretations of the same query and the ability to weight them, for example, by proportion of occurrence in a workload (See Section 5.2)
- support documents with distinct identifiers, but identical or equivalent content; ‘equivalence sets’ (See Section 4.0.1) (Our first way of encouraging novelty and diversity.)
- allow for differential weighting of queries (because some may be more popular, or business critical, or more important to the searcher.)

4. C-TEST

In this section, we introduce the C-TEST file formats, tools and other resources.

C-TEST may be downloaded from es.csiro.au/C-TEST/. A full online description is available at es.csiro.au/C-TEST/manual.html. This section provides only a brief overview of capabilities. It focuses on the features relevant to the IDR workshop.

At the heart of C-TEST are two XML document type definitions (DTDs [2]) which formalise the definition of the test and result files. These DTDs can be accessed via the website listed above.

A testfile consists of a list of query elements. Figures 6 and 7 show an example one-query testfile and results file, respectively. Commentary on the sub-elements and attributes shown in this example are given in Figure 8.

Key points to note about C-TEST files:

- the ability to specify different interpretations of a query and to weight them.
- the ability to specify groups of documents which are equivalent with respect to the information need. In other words, exposing that no extra credit should be given for returning more than one of them.
- the ability to specify the judging depth for each query.
- the ability to weight queries relative to each other.
- attributes are provided for labelling testfiles, interpretations and esets with comments explaining their provenance, meaning or significance.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<testfile name="Airline_Homepage_Finding">
  <query id="1" text="Qantas_airways" weight="1.0" depth="10">
    <interpretation comment="homepage_finding" weight="0.9">
      <eset util="10" comment="qantas_homepage">
        <docid>qantas.com
        </docid>
        <docid>www.qantas.com.au
        </docid>
        <docid>www.qantas.com/index.html
        </docid>
      </eset>
    </interpretation>
    <interpretation comment="share_price" weight="0.1">
      <eset util="10" comment="qantas_stock_exchange_listing">
        <docid>asx.com.au/companylisting=QAN
        </docid>
      </eset>
      <eset util="3" comment="QANTAS' share_information_page">
        <docid>qantas.com.au/info/about/investors/shareholderInfo
        </docid>
      </eset>
    </interpretation>
  </query>
</testfile>
```

Figure 6: Example single-query testfile (incomplete), showing two interpretations, with one and two equivalence sets

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<results label="Airways_Homepage_Finding_run_with_alpha=0_beta=2">
  <query id="1" text="Qantas_airways" weight="1.0" depth="10">
    <docid rank="1">www.qantas.com.au/</docid>
    <docid rank="2">www.qantas.com.au/regions/dyn/</docid>
    <docid rank="3">www.anzac.com/qantas/qantas.htm</docid>
    <docid rank="4">www.qantas.com/index.html</docid>
    <docid rank="5">www.quantas.com.au/</docid>
    <docid rank="6">en.wikipedia.org/wiki/Qantas</docid>
    <docid rank="7">www.airlinequality.com/Forum/qantas.htm</docid>
    <docid rank="8">www.travelmood.com/qantas.asp</docid>
    <docid rank="9">www.oneworld.com/ow/member-airlines/qantas</docid>
    <docid rank="10">www.webjet.com.au/airlines/qantas.htm</docid>
  </query>
</results>
```

Figure 7: Example single-query resultfile (incomplete)

testfile name: Can be used to document the purpose, applicability, etc. of the testfile. [text]

query id: A unique identifier whose main purpose is to enable unambiguous matching of queries. [text]

query text: The actual query. Note that ampersands and angle brackets must be XML-encoded. [text]

query weight: In many cases, all queries in a testfile will have the same weight. However, heavier weights may be assigned to certain queries, either to reflect business importance, or frequency of submission of that query. [positive real]

query comment: A human readable comment for the query as a whole. Useful for storing notes on the query during judging or holding an explanation of the query's meaning. [optional] [text]

query depth: The number of search results to be included in the evaluation. This would normally be the default maximum number of results returned by a search, typically 10. [positive integer]

interpretation An interpretation of the user's information need given this query's text. There may be multiple interpretations per query (the same effect could be achieved by repeating queries with different weights and different esets. However, it is convenient to group them together as interpretations). See earlier.

interpretation comment: A description of this interpretation. [text]

interpretation weight: The weight of this interpretation, the use of which may vary according to metric. [positive real]

eset util: This is the base utility score which will be credited if one of the document identifiers in the list is found among the results. In computing most effectiveness measures, the actual score credited will be discounted by the rank at which the document is retrieved. Once one document from an eset has been retrieved, no credit is gained by retrieving others. [positive real]

eset comment: An explanation of the documents in the eset. [optional] [text]

document identifier: each eset contains document identifiers; docid XML nodes. A document identifier may be any string which reliably identifies a document, such as a TREC DOCNO or a URL. When the documents are web documents, the use of URLs is recommended, as canonicalisation performed by the C-TEST scripts (such as removal of :80 in the host part, removal of default-page suffixes, and canonicalisation of percent-encoded special characters) reduces the need for manual listing of equivalent docids.

Figure 8: Informal explanation of the sub-elements and attributes in the C-TEST testfile example.

4.0.1 Equivalence sets

Equivalence sets (esets) are sets of document identifiers or URLs corresponding to documents which make an identical contribution to meeting the need behind the query. For example, `http://exampledomain.com/` and `http://exampledomain.com/index.html` quite likely have identical content. If they appear at ranks one and two in a result set for which they are useful, the second should not be rewarded in an evaluation, as the user gains nothing from its presence. That position in the result set would be better used for an alternative document offering new content, or even a distinct interpretation. An extreme implementation may even penalise a system for wasting a ranking position.

In C-TEST, the appearance of second or subsequent members of an eset already represented in the ranking is treated as though it were a document of no utility.

In most cases, membership of equivalence sets is unlikely to be contentious. If corresponding esets defined by two judges differ in cardinality, we believe it to be more likely due to an inadvertent miss than to a difference of opinion.

In a tuning exercise, we must assume that a testfile contains an acceptably complete set of acceptably complete esets, prior to commencing the tuning runs. In other circumstances, such as the normal TREC ad hoc submission mode, useful answers are not identified prior to running the queries. C-TEST supports this mode of operation too, by allowing testfiles whose queries have no esets.

The results file shows a text label which can be used to document how the results were obtained. It then consists of a list of query elements. Query attributes are repeated from the original testfile. A query element contains a set of docid elements. Each docid includes a rank attribute which specifies the rank at which this document was retrieved.

4.1 Representing different search scenarios in C-TEST

If a C-TEST file were prepared to tune a system used by intelligence assessors, or researchers collecting medical papers for systematic review or meta-analysis, queries in that test file may be characterised by:

- a single interpretation
- a very large judging depth
- only one level of utility

Queries submitted for the purpose of locating homepages for people or organisations, are likely to be characterised in the testfile by:

- multiple interpretations
- shallow judging depth
- only one level of utility

On the other hand, an informational search on the web may comprise queries with:

- multiple interpretations
- shallow judging depth
- multiple useful resources per interpretation, with different utility values

Later, we will show that the C-TEST format can be used to support aspectual recall and (to some extent) retrieval of elements, facets or nuggets.

4.2 C-TEST tools

A range of utilities for manipulating C-TEST files are included in the C-TEST download. They include tools for:

- generating C-TEST files:
 - from comma-separated value format
 - from TREC topics and qrels
 - by sampling from within search engine logs
- manipulating C-TEST files:
 - merging testfiles
 - dividing testfiles into test and training fractions
 - checking and tidying testfiles
 - automatically expanding esets (based on redirection and simple duplicate detection)
 - using an interactive Web judging interface to provide answers for queries in a testfile
- evaluating runs:
 - computing measures from the output of a run
 - comparing two runs, e.g. showing the queries whose results varied most across two runs
 - conducting t-tests or Wilcoxon Signed Ranks tests across a pair of runs
 - checking and tidying testfiles

There is also a directory of unsupported ‘contributed’ tools, assembled over our time using the toolkit.

These utilities are available for all to use. We welcome comments, feedback and patches. The main utilities and underlying libraries include an extensive self-test suite, to allow modifiers of the code to avoid introducing errors.

4.3 Difficulties in generating testfiles from query logs

As previously mentioned, the query log can be seen to represent the workload of a search facility over a period of time and is arguably, in most cases, the population which should be sampled. Three difficulties arise.

The first is the problem of interpreting the possible information needs behind a query. This may be easier in enterprise evaluation than on the Web because the universe of discourse is so much smaller. In our previous work, we asked the judge to think about and then externalise the possible interpretations of the query, and to record documents relevant to what were, in the judges mind, likely interpretations [13].

The second is related to the first. What to do if a judge is unable to interpret a query or can interpret it but is unable to identify any useful answer? Some possible reasons for this occurring are misdirected queries, language mismatch, spelling errors and sub-optimal publishing. In our previous work, we discarded those queries [13]. Thus, we were effectively sampling from the population of interpretable, answerable queries.

The third difficulty relates to constancy of workload. If workloads vary substantially over time, tunings performed against a sample of the workload will soon lose predictive power. It may be possible to characterise the difference in workload from one month to the next using a measure of

probability distance, such as KL-divergence or χ -square. If there is a high similarity between optimum tunings using workloads sampled at different points of time, then differences in query load are not a problem. We expect this to be the normal case in practice. However, a new testfile will certainly be needed if a website gets a makeover or changes its content management system.

We have assumed uniform sampling of logs, but there may be an argument for stratified sampling to ensure accurate representation of less common types of query (very long queries, queries with operators and so on) which might be handled differently by a search tool. We have further assumed queries should receive a weight directly proportional to frequency.

4.4 Judging

It is frequently the case that the webmaster or intranet manager within an organisation can confidently identify the key resources for at least the popular queries. In many instances, such a person is willing to spend the time to list the key answer URLs for each of the top 100 queries submitted. In other cases, such a person is willing to check lists of key resources for a set of queries, identified by an experimenter.

Often, webmasters are comfortable working with spreadsheets. Consequently, C-TEST includes scripts to convert a spreadsheet into a testfile (comma-separated value, CSV, format) in which the first column contains queries and subsequent columns contain docids (URLs) of key resources.

We have found a ‘search assisted’ judging interface is useful for completing enterprise judgements on key resources. An early version is shown in Figure 9. The user walks through queries at their own pace, and if they are unsure of relevant pages, the site’s search and a whole-of-Web search are made available to give support. The search systems are there only as a support mechanism, and often further browsing of the site or familiarity with the organisation are essential to high-quality judgements.

Click data has been identified as a valuable source of relative relevance data [9]. Click data is attractive for a number of reasons, including its potentially large volume and reflection of the true behaviour of users. However, there are many queries in a Web context having no associated clicks due to ‘brilliant success’³ or ‘abject failure’, or more clicks than intended due to ‘bounce’ [17] (where a user intends to click once, but the software registers more than one click).

4.5 Support for C-TEST in Lemur, Terrier and Zettair

C-TEST would be of little interest to researchers if the IR systems they use did not support C-TEST input and output. At the time of writing, the maintainers of Lemur⁴, Terrier⁵ and Zettair⁶ have given in-principle agreement to addition of these interfaces. We expect that the required code changes for most engines will be small.

4.6 TREC topics in C-TEST form

All of the Web track topics have been converted into C-TEST format and the converted forms are supplied with the

³i.e. the desired information is visible in the snippet without the need to click

⁴www.lemurproject.org/

⁵ir.dcs.gla.ac.uk/terrier/

⁶www.seg.rmit.edu.au/zettair/

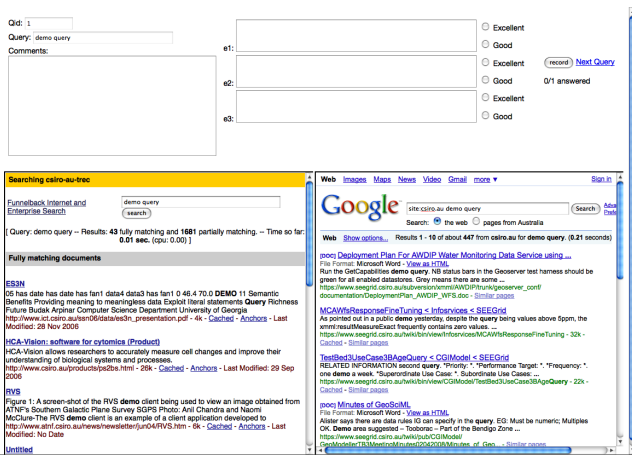


Figure 9: The assisted judging interface, showing a demonstration query. On the left, a search using the enterprise search and, on the right, a ‘site:’ restricted whole-of-Web search. At the top, edit fields allow comments to be added and docids to be entered for each eset. Only three esets and levels of relevance are available in this early tool, and interpretations are not considered.

C-TEST distribution. Conversion of other topics will depend upon demand.

5. C-TEST-SUPPORT FOR REDUNDANCY AND DIVERSITY

Here, we address some issues in our framework for evaluation; redundancy in documents retrieved and diversity in user requests.

5.1 Redundancy

As noted previously, it is possible for distinct document identifiers, such as URLs, to point to the same useful document.

Many of these cases are handled by C-TEST canonicalisation machinery; URLs in result sets and in C-TEST esets are canonicalised before comparison. When this is not the case, the C-TEST equivalence set (eset) mechanism illustrated above can be used. C-TEST supports automatic filling in of esets using redirection logs created by a crawler, or by detecting exact or near duplicates [3, 16]. Equivalent documents can be manually inserted into an eset when it is recognized that they contribute exactly the same information as an existing eset member.

Manual insertion of answers could also be used to perform evaluations for some types of aspectual recall, as studied in the TREC Interactive Track some years ago. For example, all documents describing one particular ferry sinking could be included in one eset, even if they were not really duplicates of each other.

Currently, C-TEST makes no explicit provision for partial overlap of documents, but it is possible. For example if document D_1 contains ‘nuggets’ a and b (using terminology from [5]), D_2 contains nugget c and d and D_3 contains nuggets b and c , then C-TEST can be told not to grant extra credit to a system returning all three documents over a system returning two. Further, there is no reason why the document

identifiers listed in a C-TEST file cannot reference document components such as XML elements.

For a subsequent version of C-TEST, we will consider giving utility values to individual docids rather than to the eset. This would then allow for INEX style evaluations in which the ideal element and its sub and super elements are listed in an eset with different utility values. Credit would be given for only the first member of the eset but the amount of credit would depend upon the element’s level.

5.2 Interpretations

A user seeking a document distills their information need into a query, which is then submitted to the search system. Two or more identical queries may arise from distinct information needs, however, with the search system unable to disambiguate the underlying need. The standard examples include the query ‘java’; is the user interested in a drink, a computer programming language or an Indonesian island?

For evaluation, we propose including for each query one or more ‘interpretations’. All interpretations within a query share the same query string. Each interpretation includes a series of equivalence sets and a judge’s comment, describing how the judge understood the query for the enclosed document identifiers. A judge may consider the query and documents from the viewpoint of more than one interpretation.

At present, each interpretation is linearly weighted. The weights could be devised by the judge’s view of how popular an interpretation will be, or by a more systematic means, such as an analysis of log files or side by side comparisons with one interpretation each side [17].

If the satisfaction is measured on a ratio scale, then the total satisfaction for a query is the sum weighted average satisfaction of the query’s interpretations. With a satisfaction metric, this could be used for tuning a search engine. However, it is possible, perhaps likely, that minority interpretations are drowned out by the majority, a problem similar to that described by Cooper in his counter example to the probability ranking principle [12].

An anecdotally successful approach to tuning an IR system is to attempt to maximise average satisfaction over parameter sets shown to achieve *minimal dissatisfaction*. The dissatisfaction measure could be the simple proportion of query interpretations achieving a satisfaction score of zero—no useful document in the retrieved set. Alternatively the weighting of query interpretations (query weight \times interpretation weight) could be used in calculating a weighted average.

In this way, more searchers will be satisfied to some degree, even if some are less satisfied than they might have been. Measuring satisfaction remains a challenge. As it cannot be measured directly, evaluation relies on ‘surrogate’ satisfaction measures, such as NDCG. The surrogate satisfaction measure most closely reflecting a user’s satisfaction almost certainly changes by the user and information need.

One possible future extension is to allow the author of each query interpretation to specify a preferred surrogate satisfaction measure and depth. For some interpretations of a query, such as home page finding, MRR1 may be an appropriate measure, while for another interpretation of the same query NDCG or P@20 may model the searcher’s satisfaction better. How best to aggregate scores across multiple surrogate satisfaction measures remains an open

question. A simpler alternative may be to only change the parameters of the satisfaction measure, such as the gain parameter for NDCG [8].

If desired, bolder decisions about how to use interpretations may be made for some evaluations. An example might be an evaluation of a ‘contextual’ system, where different interpretations could be run through the system with different parameters. By recording interpretations and their rationale, these decisions can be deferred.

6. CONCLUSION

Optimising the performance of search systems is an important goal for user organisations and for vendors of search technology. Meaningful optimisation must be based on unbiased sampling of the population of work submitted to the engine and on accurate modelling of users and their tasks. For search system tuning, the results and models must be encapsulated in a form which facilitates automated running with thousands of different parameter combinations.

Inevitably, there are limitations on how accurately user satisfaction can be modelled. C-TEST in its current form is not a ‘solution’ to these limitations, but rather a useful step forward, which we hope will stimulate further development. It increases the rigor and completeness of testfile specification, and provides richer capabilities for modelling average satisfaction. It does this by explicitly representing different query interpretations and importance levels, and different document utility levels, and by relaxing the assumption that relevance judgments are independent. Unlike many set-based relevance approaches, it retains usability.

We are offering the C-TEST format for use by anyone for any purpose in the hope that it might lead to more research on more meaningful batch evaluation, to further improvements in testfile design, and, ultimately, to happier searchers.

Acknowledgements

We gratefully acknowledge useful information provided by Jacques Savoy and Alexander Krumpholz. Vassilis Plachouras, Iadh Ounis, Michael Carter, Alexander Krumpholz, Ian Soboroff, Mark Sanderson, Nick Craswell, and Jacques Savoy provided valuable comments on the initial C-TEST proposals in early 2006.

7. REFERENCES

- [1] Y. Bernstein and J. Zobel. Redundant documents and search effectiveness. In *Proc. CIKM*, 2005.
- [2] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau. *Extensible Markup Language. W3C*, fifth edition, 2008.
- [3] A. Broder. On the resemblance and containment of documents. In *Proceedings Compression and Complexity of Sequences*, 1997.
- [4] C. Clarke, G. Cormack, and T. Lynam. Exploiting redundancy in question answering. In *Proc. SIGIR*, 2001.
- [5] C. Clarke, M. Kolla, G. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon. Novelty and diversity in information retrieval evaluation. In *Proc. SIGIR*, 2008.
- [6] G. Del Corso, A. Gullí, and F. Romani. Ranking a stream of news. In *Proc. WWW*, 2005.
- [7] D. K. Harman. Overview of TREC-2002 novelty track. In *Proc. TREC*, 2002.
- [8] K. Järvelin and J. Kekäläinen. IR evaluation methods for retrieving highly relevant documents. In *Proc. SIGIR*, 2000.
- [9] T. Joachims. Optimizing search engines using clickthrough data. In *Proc. KDD*, 2002.
- [10] B. Magnini, M. Negri, R. Prevete, and H. Tanev. Is it the right answer?: exploiting web redundancy for answer validation. In *Proc. ACL 2002*, 2001.
- [11] M. E. Maron and J. L. Kuhns. On relevance, probabilistic indexing and information retrieval. *J. ACM*, 7(3):216–244, 1960.
- [12] S. E. Robertson. The probability ranking principle in IR. *J. Doc.*, 33(4):296–304, 1977.
- [13] T. Rowlands, D. Hawking, and R. Sankaranarayana. Workload sampling for enterprise search evaluation. In *Proc. SIGIR*, 2007. Poster paper.
- [14] M. Sanderson. Ambiguous queries: test collections need more sense. In *Proc. SIGIR*, 2008.
- [15] S. S. Stevens. On the theory of scales of measurement. *Science*, 103(2684):677–680, 1946.
- [16] M. Theobald, J. Siddharth, and A. Paepcke. Spotsigs: robust and efficient near duplicate detection in large web collections. In *Proc. SIGIR*, 2008.
- [17] P. Thomas and D. Hawking. Evaluation by comparing result sets in context. In *Proc. CIKM*, 2006.
- [18] E. M. Voorhees. Question answering in TREC. In E. M. Voorhees and D. K. Harman, editors, *TREC: experiment and evaluation in information retrieval*. MIT Press, 2005.