Hummingbird's Fulcrum SearchServer at TREC-9

Stephen Tomlinson[1], Tom Blackwell
Hummingbird
Ottawa, Ontario, Canada

February 6, 2001

Abstract

Hummingbird submitted ranked result sets for the Main Web Task (10GB of web data) and Large Web Task (100GB) of the TREC-9 Web Track, and for Stage 2 of the TREC-9 Query Track (43 variations of 50 queries). SearchServer's Intuitive Searching produced the highest Precision@5 score (averaged over 50 web queries) of all Title-only runs submitted to the Main Web Task. SearchServer's approximate text searching and linguistic expansion each increased average precision for web queries by 5%. Enabling SearchServer's document length normalization increased average precision for web queries by 10-30% and for long queries by 100%. Squaring the importance of the inverse document frequency (relevance method 'V2:4') increased average precision in the query track by 5%. Blind query expansion decreased average precision of highly relevants for web queries by almost 15%; the same method was neutral when counting all relevants the same.

1 Introduction

Hummingbird's Fulcrum SearchServer kernel is an indexing, search and retrieval engine which runs on Windows and UNIX platforms. SearchServer, originally a product of Fulcrum Technologies, was acquired by Hummingbird in 1999. The SearchServer kernel is embedded in 5 Hummingbird products, including SearchServer, an application toolkit used for knowledge-intensive applications that require fast access to unstructured information.

The SearchServer kernel supports a variation of the Structured Query Language (SQL), called SearchSQL, which has extensions for text retrieval. Almost 200 document formats are supported, such as Word, WordPerfect, PDF and HTML. Many character sets and languages are supported, including the major European languages, Japanese, Korean, Greek and Arabic. SearchServer's Intuitive Searching algorithms were updated for version 4.0 which shipped in Fall 1999, and in subsequent releases of other products. The next major kernel release works in Unicode internally and supports many more languages [4].

2 System Description

All experiments were conducted on a single-cpu desktop system, OTWEBTREC, with a 600MHz Pentium III cpu, 512MB RAM, 186GB of external disk space on one e: partition, and running Windows NT 4.0 Service Pack 6.

For most official TREC runs, an experimental version of SearchServer 5.0 was used (a different experimental version was used for the Query Track runs in September than the Web Track runs in July and

[1] Core Technology, Research and Development, stephen.tomlinson@hummingbird.com

August). Commercial release SearchServer 4.0 was used for one Main Web Task run and one Query Track run.

3 Setup

We describe how SearchServer was used to handle the Main Web Task (10GB of web data) and Large Web Task (100GB) of the TREC-9 Web Track, and Stage 2 of the TREC-9 Query Track (1GB of news and government documents).

3.1 Data

The WT10g collection of the Main Web Task was distributed on 5 CDs. We copied the contents of each CD onto the OTWEBTREC e: drive (e:\data\wt10g\cd1 - e:\data\wt10g\cd5). The cd5\info subdirectory, containing supporting information not considered part of WT10g, was removed to ensure it wasn't indexed. The 5157 .gz files comprising WT10g were uncompressed. No further pre-processing was done on the data. Uncompressed, the 5157 files consist of 11,032,691,403 bytes (10.3GB), about 2MB each. Each file contains on average 328 "documents", for a total of 1,692,096 documents.

The WT100g collection of the Large Web Task was distributed on 2 DLT-4000 tapes. We copied the contents onto a "compressed NTFS" area of OTWEBTREC's e: drive (e:\data\compressed\wt100g). The BASE10 and BASE1 subsets are not considered part of WT100g and were stored elsewhere. We uncompressed the 50,023 files comprising WT100g from .gz format (and Windows NT internally recompressed them on the compressed NTFS drive), which took 5 hours. No further pre-processing was done on the data. Uncompressed, the 50,023 files consist of 107,828,665,842 bytes (100.4GB). Based on the change in bytes free on the drive, we estimate the files occupied about 58.8 billion bytes (54.8GB) on the compressed NTFS drive. Hence, NTFS compression saved about 46GB of space, poor compared to gzip compression (which saved 71GB), but still worthwhile. Each file contains on average 371 "documents", for a total of 18,571,671 documents. (For more information on the WT100g collection, see [3].)

Text Research Collection Volume 1, Revised March 1994, more commonly known as "TREC Disk 1", was used in Stage 2 of the Query Track, and consists of a single CD. We copied its contents to e:\data\TREC\Vol1. The various README files and the DTD directory were removed because they are not considered part of the collection. The 1265 .Z files comprising the collection were uncompressed. No further pre-processing was done on the data. Uncompressed, the 1265 files consist of 1,265,137,373 bytes (1.2GB), about 1MB each. Each file contains on average 404 "documents", for a total of 510,637 documents. (For more information on the TREC Disk 1 collection, see [10].)

3.2 Text Reader

To index and retrieve data, SearchServer requires the data to be in Fulcrum Technologies Document Format (FTDF). SearchServer includes "text readers" for converting most popular formats (e.g. Word, WordPerfect, HTML, PDF, Excel, PowerPoint, etc.) to FTDF. A special class of text readers, "expansion" text readers, can insert a row into a SearchServer table for each logical "document" inside a container, such as directory or library file. Users can also write their own text readers in C for expanding proprietary container formats and converting proprietary data formats to FTDF.

The library files of WT10g and WT100g consisted of several logical documents, each starting with a <DOC> tag and ending with a </DOC> tag. After the <DOC> tag, the unique id of the document, e.g. WTX104-B01-1, was included inside <DOCNO>..</DOCNO> tags. Other HTTP header information, such as the URL of the document, appeared inside <DOCHDR>..</DOCHDR> tags. The content of the

web document started after the </DOCHDR> tag and ended at the already-mentioned </DOC> tag. Most document's content were HTML format because only documents with mime type "text/html" were included in the collections, but on the web, some servers mislabel binaries and other file types as text/html. We made no attempt to screen out such mislabeled documents.

We wrote a custom text reader called cTREC to handle expansion of the library files of the WT10g and WT100g collections and to make a few conversions to the HTML format.

In expansion mode (/E switch), cTREC scans the library file and for each logical document determines its start offset in the file (i.e. offset of <DOC> tag), its length in bytes (i.e., distance to </DOC> tag), and extracts its document id (from inside <DOCNO>..</DOCNO> tags). SearchServer is instructed to insert a row for each logical document. The filename column (FT_SFNAME) stores the library filename. The text reader column (FT_FLIST) includes the start offset and length for the logical document (e.g. cTREC/w/100000/30000). The document id column (controllable with the /d switch), contains the document id.

In web track format translation mode (/w switch), cTREC would insert control sequences around the header to turn off indexing (i.e. from <DOC> down to the </DOCHDR> tag was not indexed). Indexing was also turned off around HTML tags, except for the content of META NAME/HTTP-EQUIV="DESCRIPTION/KEYWORDS/SUBJECT/TITLE" tags. Some entities were converted: the ones listed in the DTDs for the TREC disks 1-5, e.g. é to e, and numeric entities, e.g. é to e. Because we knew the queries were all English, we didn't try to take advantage of SearchServer's rich character support capabilities, such as accent-indexing and recognition of semantically equivalent forms of Unicode.

The library files of TREC Disk 1 also consisted of several logical documents delineated by <DOC>..</DOC> tags and identified by a <DOCNO>, so the cTREC /E switch also handled expansion of these files. When invoked without the /w or /E switch, cTREC assumes it is reading a document from TREC disks 1-5 and by default inserts control sequences to turn off indexing around all tags listed in the TREC disk 1-5 DTDs, and converts all entities listed in the DTDs. By default, cTREC also turns off indexing for data delineated by tags indicating keyword fields (namely IN, CO, G, GV, RE, MS, NS, DESCRIPT or SUBJECT tags) because the original TREC guidelines did not permit using those fields (a /k option exists for overriding this guideline). Some other tagged data is not indexed by default (nor with the /k option) because its content isn't considered helpful (for TREC Disk 1, data delineated by DOCNO, FIRST, SECOND, FILEID, NOTE, UNK, BYLINE, C, CODE-213, DOCID, NOTE, T2, T4, AUTHOR, DATE, SO, ADDRESS, AUTHOR and JOURNAL tags is not indexed by default; a longer list exists to cover the other disks). cTREC currently doesn't differentiate its tag handling by collection type; for example, the <G> tag is a keyword field in the Wall Street Journal documents, but not in the Federal Register documents, but cTREC treats it as a keyword field in both, a minor limitation. cTREC looks ahead at most 8000 bytes for an end tag when it encounters a tag indicating indexing should be turned off; if the end tag is not found, indexing is not turned off.

3.3 Indexing

WT10g was indexed in one table in most runs, created with the following SearchSQL statement:

```
CREATE SCHEMA WT10GW CREATE TABLE WT10GW
(DOCNO VARCHAR(256) 128) PERIODIC
BASEPATH 'E:\DATA' STOPFILE 'MYTREC.STP' APPROX_ZONES '32';
```

The APPROX_ZONES '32' parameter specifies that an approximate search index should be built on the external text column (32). The STOPFILE parameter specified a file containing a list of 101 stopwords to not index, including all letters and single-digit numbers. The PERIODIC parameter prevents immediate

indexing of rows at insertion time. The BASEPATH parameter specified the directory from which relative filenames of insert statements would be applied. The DOCNO column was assigned number 128 and a maximum length of 256 characters.

After creating the table, we added the string "IND:x16384;b4096;" to the wt10gw.cfg file to ensure an obscure internal dictionary limit wouldn't be encountered at index-time. This step is not necessary as of SearchServer 5.0 Beta2.

Into this table, we just inserted one row, specifying the top directory of WT10g, with this Insert statement:

```
INSERT INTO WT10GW ( FT_SFNAME, FT_FLIST )
VALUES ( 'WT10G', 'cTREC/E/d=128:s!cTREC/w/@:s');
```

To index the table, we just executed this Validate Index statement:

```
VALIDATE INDEX WT10GW VALIDATE TABLE
TEMP_FILE_SIZE 2000000000 BUFFER 256000000;
```

The VALIDATE TABLE option of the VALIDATE INDEX statement causes SearchServer to review whether the contents of container rows, such as directory rows and library files, are correctly reflected in the table. In this particular case, SearchServer initially validated the directory row by inserting each of its sub-directories and files into the table. Then SearchServer validated each of those directory and library file rows in turn, etc. Validating library file rows invoked the cTREC text reader in expansion mode to insert a row for each logical document in the library file, including its document id.

After validating the table, SearchServer indexed the table, in this case using up to 256MB of memory for sorting (as per the BUFFER parameter) and using temporary sort files of up to 2GB (as per the TEMP_FILE_SIZE parameter), to produce a dictionary of the unique words and reference file with the locations of the word occurrences (mostly unused in our experiments because we did no proximity searches nor search term highlighting). By default, SearchServer stores the original words, not just the stems.

For one of our Main Web Task runs (hum9td4), we used commercial release SearchServer 4.0, which is limited to 2GB reference files. Hence for that run we indexed WT10g in 2 tables. The first table contained CD1, CD2, and the first 4 directories of CD5 (WTX097-WTX100). The second table contained CD3, CD4 and the last 4 directories of CD5 (WTX101-WTX104).

Because of various internal limits in the experimental SearchServer version used for WT100g, we indexed WT100g in 12 tables (more than proved to be necessary). No approximate search index was built; however, some Large Web Task runs re-used the approximate search index of WT10g as a spell-corrector.

For the Query Track, we indexed TREC Disk 1 three times, once filtering keywords as per the traditional TREC guidelines, a second time with keyword fields included in the index, and a third time using commercial release SearchServer 4.0 (including keywords).

4 Search Techniques

For the Main Web Task, the 50 "topics" were in a file called "topics.451-500". The topics were numbered from 451-500, and each contained a Title (which was an actual web query taken from a search engine log), a Description (NIST's interpretation of the query, with spelling and grammar errors fixed), and a Narrative (a more detailed set of guidelines for what a relevant document should or should not contain).

For the Large Web Task, the 10000 web queries were in a file called "queries_10000". Queries were numbered from 20001-30000. There was no separate Title, Description or Narrative, just the original web queries, one per line.

For Stage 2 of the Query Track, there were 43 separate files of 50 queries each (variants of TREC topics 51-100).

We modified the example stsample.c program included with SearchServer to parse the TREC topics files, construct and execute corresponding SearchSQL queries, fetch the top 1000 or top 20 rows, and write out the rows in the results format requested by NIST. (The modified stsample.c was called QueryToRankings.c.)

SELECT statements were issued with the SQLExecDirect api call. Fetches were done with SQLFetch (typically 1000 SQLFetch calls per query in the Main Web Task and Query Track, and 20 SQLFetch calls per query in the Large Web Task).

4.1 Intuitive Searching

All queries used SearchServer's Intuitive Searching, i.e. the IS_ABOUT predicate of SearchSQL, which accepts unstructured text. For example, for topic 451 of the Main Web Task, the Title was "What is a Bengals cat?". A corresponding SearchSQL query would be:

```
SELECT RELEVANCE('V2:3') AS REL, DOCNO
FROM WT10GW
WHERE FT_TEXT IS_ABOUT 'What is a Bengals cat?'
ORDER BY REL DESC;
```

This query would create a working table with the 2 columns named in the SELECT clause, a REL column containing the relevance value of the row for the query, and a DOCNO column containing the document's identifier. The ORDER BY clause specifies that the most relevant rows should be listed first. Typically a statement such as "SET MAX_SEARCH_ROWS 1000" was previously executed so that the working table would contain at most 1000 rows. In cases where the data was indexed in more than one table, the FROM clause would specify a UNION of the tables, e.g. SearchServer 4.0 queries contained "FROM WT10GW1 UNION WT10GW2".

Our QueryToRankings program removed a short list of words from the given topics before presenting them to SearchServer: "documents", "document", "items", "item", "relevant". This was originally done for internal TREC-5 experiments based on the TREC-5 topics frequently containing these words (e.g. "A relevant item will mention..."). It was found to make almost no difference to the scores whether these words were excluded or not, so we didn't bother to expand the list further for the TREC-9 Main Web Task. For the Large Web Task, in which most queries were known to be phrased as questions, we additionally removed the words "do", "does", "find", "how", "me", "show", "tell", "what" and "why".

4.1.1 Secondary Term Selection

The IS_ABOUT predicate by default expands each word to a "superterm" comprising all the linguistic variants of the term, e.g. "run" is added for "ran" (linguistic expansion can be disabled with the VECTOR_GENERATOR parameter). Some of these superterms may be subsequently discarded when searching the table (secondary term selection). For example, the RELEVANCE_METHOD setting has an optional document frequency parameter for discarding all terms which occur in more than a specified percentage of the rows (based on the most frequently occurring variant of the term). Secondary term selection improves performance and prevents highlighting of unimportant terms.

In the SearchServer 5.0 web track runs, we experimented with a different term selection approach based on an estimate of how many rows the terms would bring into the search and which involved a different formula for term importance which incorporated the vector length. In the SearchServer 4.0 runs, in which the experimental approach wasn't available, a document frequency cutoff of 15% was normally used, because that cutoff in past experiments didn't hurt quality, but significantly improved performance.

4.1.2 Statistical Relevance Ranking

To calculate a relevance value for a row of a table with respect to the vector of terms (actually, superterms) resulting from secondary term selection, the inverse document frequency of the term and the number of occurrences of the term in the row (term frequency) are determined from the index. The length of the row (based on the number of indexed characters in all columns of the row, which is typically dominated by the external document), is optionally incorporated, and the number of occurrences of the term in the vector is also used. The full details of synthesizing this information into a relevance value are proprietary, but draws from [7] (particularly the Okapi approach to term frequency dampening) and [9]. SearchServer's relevance values are always an integer in the range 0 to 1000.

SearchServer's RELEVANCE_METHOD setting can be used to optionally square the importance of the inverse document frequency (by choosing a RELEVANCE_METHOD of 'V2:4' instead of 'V2:3'). Experiments on past TREC ad hoc topics found that V2:4 often worked better than V2:3, but past TREC ad hoc topics didn't contain spelling errors, which could be over-emphasized when squaring the idfs.

SearchServer's RELEVANCE_DLEN_IMP parameter controls the importance of document length (scale of 0 to 1000) to the ranking. We found 200 worked best on TREC-8 small web experiments and used 200 for most submitted TREC-9 runs.

4.2 Approximate Text Searching

The Title-only queries of the Main Web Task and the queries of the Large Web Task were unedited web queries from search engine logs which appeared to sometimes contain spelling errors; for example, a query containing "vanila ice creem" probably meant to say "vanilla ice cream".

SearchServer's approximate text searching is based on edit distance, also known as the Levenshtein distance, which is the minimum number of insertions, deletions and/or replacements needed to transform one pattern into another. SearchServer's approximate text searching is fast for error ratios up to at least one-third, i.e. when the allowed edit distance is one-third the length of the search term. An excellent overview of approximate text searching techniques may be found in [6].

We experimented with using SearchServer's approximate text searching to fix some spelling errors. The first step was to look up the closest matches of each term in the web query by increasing edit distance and decreasing number of rows containing the term. For example, the SearchSQL to find the closest matches to "vanila" is

```
SELECT TERM, FT_DISTANCE(TERM) AS NUMERRORS,
  MAX(ROW_COUNT) AS NUMROWS
FROM SEARCH_TERMS
WHERE TABLE_NAME CONTAINS 'WT10GW'
 AND COLUMN_NAME CONTAINS 'FT_TEXT'
 AND TERM CONTAINS 'vanila' BEST_MATCHES 1000
GROUP BY TERM
ORDER BY NUMERRORS, NUMROWS DESC;
```

The results in the case of "vanila" were

```
('VANILA',  0,  8)
('VANILLA',  1,  2427)
('MANILA',  1,  1763)
('VANIA',  1,  47)
('VANITA',  1,  21)
('VAXILA',  1,  11)
('DANILA',  1,  10)
('VANINA',  1,  9)
('VANNILA',  1,  5)
...
```

We used the default error ratio of 34% for all approximate searches, e.g. 2 errors were allowed in the 6-letter vanila query. The WT10G collection contained 5,792,772 distinct words. SearchServer used an in-memory approximate search index to substantially reduce the time needed to find the close matches.

If the original term appeared in fewer than 10 rows, then the closest matches were added to the query until the sum of the row counts was 10 or more, with the following adjustments:

- On the first pass through the list of close matches, only matches with the same Soundex code [5] were considered. For example, "vanilla" has the same Soundex code as "vanila", but "manila" does not. If the Soundex matches didn't sum to 10 or more rows, then the closest non-Soundex matches were added until the sum was 10 or more rows. (Note: Soundex was implemented in the QueryToRankings program, not SearchServer.)

- If the row sum was still less than 10 after adding all close matches, the last character of the term was dropped, and the process repeated. For example, after "vanila", the next search would be for "vanil", then "vani", etc.

In the case of "vanila", the term occurred in just 8 rows, fewer than 10 , so the next term, "vanilla" was considered. It was a Soundex match, so it was added to the query, and adding its row count of 2427 exceeded the sum requirement, so the search for more close matches ended.

In the case of "creem", the heuristic didn't work because "creem" appeared in 43 rows, more than our arbitrary parameter of 10. In retrospect, we probably should have added the first Soundex match which occurred in more rows than the original term (if any), making an arbitrary parameter unnecessary. The closest match to "creem" is "creek" (17,521 rows), which Soundex would filter out. The next closest match is "cream" (10,692 rows), which is the term we wanted. Also, it may be better to just sort close matches by decreasing number of rows and not differentiate by edit distance. This change would have properly handled the case of "australai" (4 rows); our heuristic generated "australi" (99 rows, 1 difference), but probably the best match was "australia" (75,297 rows, 2 differences).

The truncation heuristic was an attempt to deal with words stuck together, e.g. "londonengland", but it wasn't very successful. A better solution may be to include phrases in the word list.

We kept the original terms in the query, e.g. the query "vanila ice cream" was changed to "vanila ice cream vanilla". A downside of this approach is that the ranking algorithm treats vanila and vanilla as separate terms, hence over-weighting documents which happen to contain both terms. If we integrated this approach into SearchServer, we could treat the terms as variants of one term, like we do for linguistic variations of the term.

4.3 Row Expansion

In past TRECs, "query expansion" was considered necessary to produce top results [11]. We experimented with using row expansion to indirectly expand the query in 2 of our Main Web Task submissions. The approach was built on top of SearchServer. An optimized version may be implemented inside SearchServer in a future release.

After running the initial query (possibly already expanded by approximate text searching in the Title-only case), we retrieved the top 1000 rows (unless SearchServer returned fewer, which sometimes happened for Title-only queries). For each of the top 5 rows, we asked SearchServer's Intuitive Searching to "find more rows like this" (we call these row expansion queries), retrieving the top 1000 rows. We then combined the relevance values from each of the 6 result sets, giving a weight of 5 to the original query results, and weights of 1 to each of the 5 row expansion results. We did not include any negative information. Mathematically, this approach works out to be similar to Rocchio expansion. (A detailed description of a good Rocchio feedback technique is in [1].)

We always used the same parameters in row expansion queries as were used in the initial query, e.g. the same document length importance. We used the top 5 rows because in experiments on the TREC-8 small web we found somewhere from 3 to 8 rows usually gave best results.

5 Results

The evaluation measures are explained in an appendix of the conference proceedings. Briefly: Precision is the percentage of retrieved documents which are relevant. Precision@*n* is the precision after *n* documents have been retrieved. Average precision for a topic is the average of the precision after each relevant document is retrieved (using zero as the precision for relevant documents which are not retrieved). Recall is the percentage of relevant documents which have been retrieved. Interpolated precision at a particular recall level for a topic is the maximum precision achieved for the topic at that or any higher recall level. For a set of topics, the measure is the average of the measure for each topic (i.e. all topics are weighted equally).

Below we present an analysis of our results, including results of several unofficial "diagnostic" runs. Table 1 summarizes the "official" web track runs we submitted for judging in August 2000:

Run	hum9te	hum9tde	hum9td4	hum9tdn	hum9w1	hum9w2	hum9w3
Task	Main	Main	Main	Main	Large	Large	Large
Topic Fields	T-only	T+D	T+D	T+D+N	web	web	web
SearchServer Ver.	5.0Tr1[2]	5.0Tr1	4.0	5.0Tr1	5.0Tr1	5.0Tr1	5.0Tr1
Approx. Search	Y	N	N	N	Y	N	Y
Row Expansion	Y	Y	N	N	N	N	N
Linguistic Exp.	Y	Y	Y	Y	Y	Y	N
REL...METHOD	V2:3	V2:3	V2:3:15	V2:4	V2:3	V2:3	V2:3
REL...DLEN_IMP	200	200	200	200	200	200	0
REL...AVG_DLEN	10000	10000	10000	10000	10000	10000	n/a
Exp'l Sec. Term Sel.	Y	Y	N	Y	Y	Y	Y

Table 1: Summary of Runs submitted for the TREC-9 Web Track

[2] Build 5.0.0.61 with experimental changes for TREC

5.1 Main Web Task

The Main Web Task was to run 50 queries against 10GB of web data and submit a list of the top-1000 ranked documents to NIST for judging.

Topics were broken into 3 categories: automatic runs which used only the original Excite web query (called Title-only runs), automatic runs which used some other part of the topic statement (called Full Topic runs), and manual runs. We did not submit any manual runs, just automatic runs.

NIST produced a "qrels" file: a list of documents judged to be highly relevant, relevant or not relevant for each topic. From these, the scores were calculated with Chris Buckley's *trec_eval* program, which counts all relevants the same, including highly relevants. To produce scores which just counted highly relevants as relevant, we ran *trec_eval* a 2nd time on a modified version of the qrels file which had the ordinary relevants filtered out, then multiplied by 50/46 (in 4 of the 50 topics, there were no highly relevants). Hence the scores focused on highly relevants are averaged over just 46 topics.

The medians were derived from the statistics provided in the draft conference proceedings at the conference, which counted all relevants the same. For the Title-only category, the medians are the 9th-highest score of the 18 groups, just counting the highest score from each group in each measure. For the Full Topic category, the median is the 10th-highest score of the 19 groups.

5.1.1 Title-only runs

Table 2 shows Title-only runs produced with the experimental SearchServer 5.0 in July 2000. The only differences between these runs were the relevance method (V2:3 or V2:4) and whether or not row expansion post-processing was applied. Run 2b was the official "hum9te" run, which had the highest Precision@5 score and highest interpolated precision at the 10%, 20%, 30% and 40% recall levels of the 40 submitted Title-only runs from 18 groups:

SearchServer run	AvgP	P@5	P@10	P@20	Rec0	Rec30	AvgH	H@5	H0
2a: V2:3[3]	0.1949	32.0%	26.2%	22.0%	0.5223	0.2696	0.1948	15.7%	0.3264
2b: V2:3 + exp	0.1970	32.4%	25.4%	21.5%	0.4802	0.2808	0.1703	13.5%	0.2732
2c: V2:4	0.1931	29.6%	25.0%	21.5%	0.5170	0.2674	0.2078	15.7%	0.3441
2d: V2:4 + exp	0.1909	29.6%	23.8%	21.1%	0.4756	0.2774	0.1778	14.8%	0.2618
Median (18 grps)	0.1464	21.6%	21.2%	17.4%	0.4015	0.1993	n/a	n/a	n/a

Table 2: Precision of Title-only runs

Glossary:
AvgP: Average Precision (defined above)
P@5, P@10, P@20: Precision after 5, 10 and 20 documents retrieved, respectively
Rec0, Rec30: Interpolated Precision at 0% and 30% Recall, respectively
AvgH: Average Precision just counting Highly Relevants as relevant
H@5: P@5 just counting Highly Relevants as relevant
H0: Rec0 just counting Highly Relevants as relevant

[3] Scores for diagnostic runs may differ slightly from those given in the notebook paper because, for this paper, ties in relevance values were broken according to SearchServer's ordering in the result list.

Impact of Relevance Method (compare 2a to 2c, or 2b to 2d): V2:3 was modestly better at finding relevant documents (columns AvgP through Rec30), but V2:4 was modestly better at finding highly relevant documents (columns AvgH through H0, except in 1 H0 case).

Impact of Row Expansion (compare 2a to 2b, or 2c to 2d): Our experimental row expansion post-processing made little difference for relevants, but hurt the scores when focusing on highly relevants. Perhaps the finding of past TRECs that query expansion is usually necessary for top results is not valid when just focusing on highly relevants. However, the results of many groups will have to be considered, not just ours.

To measure the impact of approximate text searching and linguistic expansion, Table 3 shows runs which were done in January 2001 with a more recent SearchServer build (5.0.500.14). This version contained a new linguistic package and did not use the experimental secondary term selection (instead, terms in more than 15% of documents were discarded (relevance method V2:3:15)). No row expansion was done, and document length importance was increased to 750:

SearchServer Run	AvgP	P@5	P@10	P@20	Rec0	Rec30	AvgH	H@5	H0
3a: ling only	0.1919	30.4%	25.6%	21.5%	0.5265	0.2686	0.2343	15.7%	0.3548
3b: apx, ling	0.2019	32.0%	27.2%	22.9%	0.5516	0.2769	0.2509	16.5%	0.3647
3c: apx only	0.1914	32.4%	28.0%	22.8%	0.5586	0.2693	0.2273	17.0%	0.3898
3d: neither	0.1805	30.4%	26.4%	21.6%	0.5258	0.2562	0.2089	15.7%	0.3535

Table 3: Impact of Approximate Text Searching and Linguistic Expansion (Title-only runs)

Impact of Approximate Text Searching (compare 3a to 3b, or 3d to 3c): The spell-correction heuristics increased most precision scores by just 1-2 points. Almost all of the improvement was in 2 topics: 487 ("angioplast7", for which "angioplasty" was added) and 463 ("tartin", for which "tartan" was added). 2 topics became a little worse, 481 and 495, because "1920" was unnecessarily added for "1920's", apparently over-weighting that term.

Impact of Linguistic Expansion (compare 3c to 3b, or 3d to 3a): Linguistic expansion improved average precision, but slightly lowered Precision@10. In average precision, topic 469 was helped ("steinbach nutcracker") as was 490 ("motorcycle safety helmets"), but topic 492 was hurt ("us savings bonds") as was 458 ("fasting"). Note that all topics were English. SearchServer's linguistic expansion is likely to be more useful for languages with more noun forms, such as German and Finnish.

Table 4 shows additional runs just varying in the setting of document length importance (RELEVANCE_DLEN_IMP parameter). These runs used build 5.0.500.14, approximate text searching and linguistic expansion were both on, and the relevance method was 'V2:4:15':

DLen Importance	AvgP	P@5	P@10	P@20	Rec0	Rec30	AvgH	H@5	H0
4a: 0	0.1595	26.0%	21.2%	17.7%	0.4393	0.2222	0.1521	10.9%	0.2554
4b: 250	0.1908	29.6%	24.2%	21.2%	0.4960	0.2677	0.1926	14.8%	0.3084
4c: 500	0.2050	31.2%	26.0%	21.8%	0.5410	0.2828	0.2282	16.1%	0.3427
4d: 750	0.1992	30.0%	24.6%	21.7%	0.5539	0.2788	0.2528	16.1%	0.3878
4e: 1000	0.1744	27.6%	20.8%	18.7%	0.4892	0.2341	0.2350	16.1%	0.3318

Table 4: Impact of Document Length Normalization (Title-only runs)

Impact of Document Length Normalization: Ignoring document length (row 4a) hurt all scores; average precision was 10-30% higher in the other rows. The impact on highly relevants was even larger. Generally, we find that settings of 200 or higher all work pretty well for average precision, but higher settings appear to be better for finding highly relevants. The setting of 750 was probably the best overall in Table 4. See Table 6 for another document length experiment.

5.1.2 Full Topic runs

The other category for automatic runs were runs which included any part of the topic besides the Title field. The median precision scores were higher for this category, as were SearchServer's scores, which makes sense because the queries had the more detailed Description and/or Narrative fields included.

Table 5 shows Full Topic runs produced in July 2000. The differences between these runs were the relevance method (V2:3 or V2:4), whether or not row expansion post-processing was applied, whether or not commercial release SearchServer 4.0 was used, and whether or not the Narrative was included. Runs 5b, 5c and 5h were submitted (official runs "hum9tde", "hum9td4" and "hum9tdn" respectively). All runs were above the median in average precision:

SearchServer run	AvgP	P@5	P@10	P@20	Rec0	Rec30	AvgH	H@5	H0
5a: T+D, V2:3	0.2374	39.2%	30.8%	25.3%	0.6202	0.3336	0.2397	17.0%	0.3930
5b: 5a + exp	0.2217	37.2%	29.4%	24.0%	0.5217	0.3082	0.1783	14.8%	0.2722
5c: SS4, V2:3:15	0.2115	37.6%	30.8%	24.8%	0.5990	0.3051	0.2053	15.2%	0.3628
5d: 5a + Narr	0.2184	39.2%	34.0%	26.3%	0.6059	0.3201	0.2005	15.2%	0.3313
5e: T+D, V2:4	0.2347	36.0%	30.6%	25.2%	0.5617	0.3190	0.2372	17.8%	0.3635
5f: 5e + exp	0.2228	34.0%	28.6%	24.9%	0.4895	0.3114	0.1862	14.3%	0.2675
5g: SS4, V2:4:15	0.2380	36.0%	30.8%	25.0%	0.5735	0.3197	0.2301	17.4%	0.3659
5h: 5e + Narr	0.2335	42.0%	35.2%	27.4%	0.6391	0.3341	0.2158	16.5%	0.3790
Median (19 grps)	0.1948	36.0%	31.4%	24.6%	0.6029	0.2692	n/a	n/a	n/a

Table 5: Precision of Full Topic runs

Impact of Row Expansion (compare 5a to 5b, or 5e to 5f): Row expansion post-processing hurt all scores, especially for highly relevants, as in the Title-only case.

Impact of Relevance Method (compare 5a to 5e, 5b to 5f, 5c to 5g, or 5d to 5h): More often than not, V2:4 was a little better, including for highly relevants, but it made little difference.

Impact of including the Narrative (compare 5a to 5d, or 5e to 5h): Including the Narrative hurt average precision scores. It increased relevants early in the result list, but not highly relevants.

Difference from SearchServer 4.0 (compare 5a to 5c, or 5e to 5g): SearchServer 4.0, which split the data into 2 tables and used the simpler secondary term selection, produced scores which were a little lower than SearchServer 5.0's with V2:3, and about the same with V2:4.

Table 6 shows a dramatic result when re-doing the runs of Table 4 (RELEVANCE_DLEN_IMP experiment) with the Description and Narrative included:

DLen Importance	AvgP	P@5	P@10	P@20	Rec0	Rec30	AvgH	H@5	H0
6a: 0	0.1136	20.8%	18.4%	15.9%	0.3684	0.1672	0.1077	7.8%	0.1846
6b: 250	0.2233	40.0%	34.8%	28.0%	0.6320	0.3219	0.2007	16.5%	0.3628
6c: 500	0.2435	44.8%	35.2%	29.5%	0.6833	0.3330	0.2447	19.1%	0.4264
6d: 750	0.2569	43.2%	36.8%	30.1%	0.6958	0.3384	0.2843	20.9%	0.4845
6e: 1000	0.2454	42.0%	36.6%	28.0%	0.6894	0.3388	0.2908	20.9%	0.4825

Table 6: Impact of Document Length Normalization (T+D+N runs)

Impact of Document Length Normalization: Ignoring the document length (row 6a) significantly hurt all scores; average precision was about 100% higher in the other rows. Many irrelevant long documents (e.g. 500KB or more) were brought into the search result by the numerous, relatively unimportant terms in the long queries when there was no document length adjustment. It appears that even a low setting, such as 250, was enough to overcome the issue. As in the Title-only case, the impact was even larger for highly relevants, and higher settings were better. Again, probably 750 was the best setting overall in Table 6.

5.2 Large Web Task

Results of our Large Web Task runs are summarized in Table 7. Only 84 of the 10000 queries were judged, and only the top 10 documents submitted for each query were judged:

SearchServer Run	Reciprocal Rank of First Satisfactory	Precision@1	Precision@5	Precision@10
hum9w1	0.4381	30.95%	32.62%	32.50%
hum9w2	0.4262	30.95%	31.43%	31.67%
hum9w3	0.4174	28.57%	30.24%	29.40%

Table 7: Precision of Large Web Task runs

The use of approximate text searching for handling misspelled terms (used in hum9w1 but not hum9w2, the only difference) improved most precision scores by 1 point. The benefit primarily came from query 28616 ("where can i find a good deal on a mothermoard") for which the term "motherboard" was helpfully added.

Turning off document length normalization and linguistic expansion (as done in hum9w3, the only differences from hum9w1) lowered precision scores by just 2-3 points. This result is in line with what one would expect from the Main Web Title-only findings for Precision@5 and Precision@10, which suggest that removing the document length adjustment hurt 3-4 points, but disabling linguistics helped 0-1 points. Unfortunately, the pool of documents submitted per topic is too small for this task for us to run meaningful experiments on isolated factors after the fact like we could for the Main Web, e.g. perhaps the individual impact of document length and linguistics is actually higher in this task.

We divided WT100g into 12 tables, each with their own set of inverse document frequencies. This need not lower the scores: AT&T found that Precision@10 scores were actually a little higher when they split WT100g into 20 tables in TREC-8 (see [8]). However, our preliminary experiments with global idfs suggest that the table-splitting may have cost us several points of precision; e.g. with global idfs, we get 30% in precision@10 with 30% unjudged, and many of the unjudged appear to be satisfactory. Our experimental secondary term selection was set more aggressively for this task than in the Main Web, and there would have been some inconsistencies in the terms discarded for different tables in our official runs.

For query 27028 ("s3 patches"), we regret that 's' and '3' were stop words. Perhaps we should enable SearchServer's option of parsing numbers as if they were letters.

5.3 Query Track

The Query Track is for evaluating not just retrieval systems, but the effect of query variations on such systems. For background on this track, see [2].

In Stage 1 of the Query Track, participants created variations of old TREC topics 51-100, including very short queries (2-3 words), natural language sentence queries, and queries based on reading system results without consulting the original topics. In all, 43 sets of 50 queries were produced by 6 different groups. We did not submit any queries for Stage 1.

In Stage 2, all groups, including those which did not submit queries, were asked to run all the query sets on their systems. The more systems, the more reliable the conclusions about varying queries. We contributed 7 runs. The overall average precision scores for each of these runs (averaged over all 43*50 queries) are in Table 8:

Run	AvgP	Experiment (i.e. what was different from baseline)
humB*	0.1732	baseline
humK*	0.1713	keyword fields were not indexed (/k option of cTREC text reader was not used, see section 3.2)
humD*	0.1771[4]	document length importance was set low (RELEVANCE_DLEN_IMP was set to 200 (baseline was 750))
humV*	0.1648	inverse document frequency not squared (RELEVANCE_METHOD was 'V2:3:15' (baseline was 'V2:4:15'))
humA*	0.1741	approximate text searching added fixes for spelling errors (algorithm of section 4.2 except the table used to index TREC Disk 1 with keywords was used)
hum4*	0.1713	SearchServer 4.0 was used (baseline used experimental SS 5.0 which contained a new linguistic expansion package which was known to still have a few glitches)
humI*	0.1736	terms in more than 15% of rows not discarded (RELEVANCE_METHOD was 'V2:4:100' (baseline was 'V2:4:15'))

Table 8: Average Precision of Query Track runs

These results suggest that excluding keyword fields makes little difference. Decreasing the document length importance was modestly helpful. SearchServer's relevance method 'V2:4', which squared the importance of the inverse document frequency, produced modestly better results. The attempt at handling spelling errors was of only minor benefit, though probably relatively few queries contained spelling errors (compared to the web queries). The experimental SearchServer 5.0 scores were slightly higher than those of SearchServer 4.0, despite some known glitches in the new linguistic package, such as expanding "in" to "Indiana" (since ironed out).

Perhaps the most interesting result is that excluding terms which occur in more than 15% of the documents, which helps search-time performance, didn't decrease average precision significantly. This result is consistent with other experiments we have done, but differs from the finding reported in Managing Gigabytes that discarding frequent terms "greatly reduces retrieval effectiveness" [12] (page 427).

[4] We received corrections to the humD and humV results after submitting the notebook paper.

References

[1] Chris Buckley, Amit Singhal and Mandar Mitra. Using Query Zoning and Correlation Within SMART: TREC 5. In E.M. Voorhees and D.K. Harman, editors, *Proceedings of the Fifth Text REtrieval Conference (TREC-5).* NIST Special Publication 500-238. http://trec.nist.gov/pubs/trec5/t5_proceedings.html

[2] Chris Buckley and Janet Walz. The TREC-8 Query Track. In E.M. Voorhees and D.K. Harman, editors, *Proceedings of the Eighth Text REtrieval Conference (TREC-8).* NIST Special Publication 500-246. http://trec.nist.gov/pubs/trec8/t8_proceedings.html

[3] David Hawking, Nick Craswell and Paul Thistlewaite. Overview of the TREC-7 Very Large Collection Track. In E.M. Voorhees and D.K. Harman, editors, *Proceedings of the Seventh Text REtrieval Conference (TREC-7).* NIST Special Publication 500-242. http://trec.nist.gov/pubs/trec7/t7_proceedings.html

[4] Andrew Hodgson. Converting the Fulcrum Search Engine to Unicode. In *Sixteenth International Unicode Conference*, Amsterdam, The Netherlands, March 2000.

[5] Donald E. Knuth. The Art of Computer Programming, Vol. 3: Sorting & Searching, 2nd edition Revised, January 1998. Addison Wesley Longman.

[6] Gonzalo Navarro. Approximate Text Searching, PhD Thesis, Dept. of Computer Science, University of Chile, December 1998.

[7] S.E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford. (City University.) Okapi at TREC-3. In D.K. Harman, editor, *Overview of the Third Text REtrieval Conference (TREC-3).* NIST Special Publication 500-226. http://trec.nist.gov/pubs/trec3/t3_proceedings.html

[8] Amit Singhal, Steve Abney, Michiel Bacchiani, Michael Collins, Donald Hindle and Fernando Pereira. AT&T at TREC-8. In E.M. Voorhees and D.K. Harman, editors, *Proceedings of the Eighth Text REtrieval Conference (TREC-8).* NIST Special Publication 500-246.

[9] Amit Singhal, John Choi, Donald Hindle, David Lewis and Fernando Pereira. AT&T at TREC-7. In E.M. Voorhees and D.K. Harman, editors, *Proceedings of the Seventh Text REtrieval Conference (TREC-7).* NIST Special Publication 500-242. http://trec.nist.gov/pubs/trec7/t7_proceedings.html

[10] Ellen M. Voorhees and Donna Harman. Overview of the Eighth Text REtrieval Conference (TREC-8). In E.M. Voorhees and D.K. Harman, editors, *Proceedings of the Eighth Text REtrieval Conference (TREC-8).* NIST Special Publication 500-246. http://trec.nist.gov/pubs/trec8/t8_proceedings.html

[11] Ellen M. Voorhees and Donna Harman. Overview of the Seventh Text REtrieval Conference (TREC-7). In E.M. Voorhees and D.K. Harman, editors, *Proceedings of the Seventh Text REtrieval Conference (TREC-7).* NIST Special Publication 500-242. http://trec.nist.gov/pubs/trec7/t7_proceedings.html

[12] Ian H. Witten, Alistair Moffat and Timothy C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images.* 2nd edition, 1999. Morgan Kaufmann Publishers.

Hummingbird, Fulcrum, SearchServer, SearchSQL and Intuitive Searching are the intellectual property of Hummingbird Ltd. All other company and product names are trademarks of their respective owners.