Hummingbird's Fulcrum SearchServer at CLEF 2001

Stephen Tomlinson[1]
Hummingbird
Ottawa, Ontario, Canada

August 4, 2001

Abstract

Hummingbird submitted ranked result sets for all 5 Monolingual Information Retrieval tasks (German, French, Italian, Spanish and Dutch) of the Cross-Language Evaluation Forum (CLEF) 2001. For each language, at least one SearchServer run had more average precision scores above the median than below. The submitted German, Dutch and Spanish runs, compared to the medians in average precision by topic, had a significance level of less than 1% (by the sign test), which is statistically significant. SearchServer's linguistic expansion was found to boost all investigated precision scores, including average precision and Precision@5, for all 6 investigated languages (German, French, Italian, Spanish, Dutch and English). Enabling linguistic expansion was found to increase average precision by 43% in German, 30% in Dutch, 18% in French, 16% in Italian, 12% in Spanish and 12% in English. In terms of the number of topics of higher and lower average precision, the German, Dutch and Spanish runs with linguistics enabled, compared to corresponding runs with linguistics disabled, had a significance level of less than 1% (by the sign test).

1 Introduction

Hummingbird's Fulcrum SearchServer kernel is an indexing, search and retrieval engine for embedding in Windows and UNIX information applications. SearchServer, originally a product of Fulcrum Technologies, was acquired by Hummingbird in 1999. Fulcrum, founded in 1983 in Ottawa, Canada, produced the first commercial application program interface (API) for writing information retrieval applications, Fulcrum Ful/Text. 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.

SearchServer supports a variation of the Structured Query Language (SQL), called SearchSQL, which has extensions for text retrieval. SearchServer conforms to subsets of the Open Database Connectivity (ODBC) interface for C programming language applications and the Java Database Connectivity (JDBC) interface for Java applications. Almost 200 document formats are supported, such as Word, WordPerfect, Excel, PowerPoint, PDF and HTML. Many character sets and languages are supported, including the major European languages, Japanese, Korean, Chinese, 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. SearchServer 5.0, which shipped in Spring 2001, works in Unicode internally [3] and contains improved natural language processing technology, particularly for languages with many compound words, such as German, Dutch and Finnish.

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 the official CLEF runs, internal development builds of SearchServer 5.0 were used (5.0.501.115

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

plus some experimental changes motivated by tests on the CLEF 2000 collections). For the diagnostic runs, internal build 5.0.504.157 was used.

3 Setup

We describe how SearchServer was used to handle the 5 Monolingual tasks of CLEF 2001.

3.1 Data

The CLEF 2001 collections consisted of tagged (SGML-formatted) news articles (mostly from 1994) in 6 different languages: German, French, Italian, Spanish, Dutch and English. The articles (herein called "documents" or "logical documents") were combined into "library files" of typically a few hundred logical documents each. The collections were made available in compressed tar archives on the Internet. We downloaded the archives, uncompressed them, untarred them and removed any support files (e.g. dtd files) which weren't considered part of the collection. The library files were stored in 6 subdirectories of e:\data\CLEF (German, French, Italian, Spanish, Dutch and English). No further pre-processing was done on the data, i.e. SearchServer indexed the library files directly. Operations such as identifying logical documents and enforcing the restrictions on fields permitted for indexing were handled by the text reader (described below).

Language	Text Size (uncompressed)	Number of Documents	Number of Library Files
German	555,285,140 bytes (530 MB)	225,371	520
French	253,528,734 bytes (242MB)	87,191	682
Italian	290,771,116 bytes (277MB)	108,578	721
Spanish	544,347,121 bytes (519MB)	215,738	364
Dutch	558,560,087 bytes (533MB)	190,604	1228
English	441,048,231 bytes (421MB)	113,005	365

Table 1: Sizes of CLEF 2001 Collections

For more information on the CLEF collections, see the CLEF web site [1].

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, 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 the CLEF 2001 collections 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. SDA.940101.0001, was included inside <DOCNO>..</DOCNO> tags. The custom text reader called cTREC, originally written for handling TREC collections [6], handled expansion of the library files of the CLEF collections and was extended to support the CLEF guidelines of only indexing specific fields of specific documents.

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.

nti/t=Win_1252_UCS2:cTREC/C/100000/30000). The document id column (controllable with the /d switch), contains the document id.

In CLEF format translation mode (/C switch), cTREC inserts a control sequence to turn off indexing at the beginning of the document. cTREC inserts a control sequence to enable indexing after tags for which indexing is permitted (e.g. after <TEXT> in a Frankfurter Rundschau document), and inserts control sequences to disable indexing just before its corresponding closing tag (e.g. </TEXT>). The document type is determined from the prefix of the DOCNO (e.g. all Frankfurter Rundschau document ids had "FR" as a prefix). The entities described in the DTD files were also converted, e.g. "=" was converted to the equal sign "=".

The documents were assumed to be in the Latin-1 character set, the code page which, for example, assigns e-acute (é) hexadecimal 0xe9 or decimal 233. cTREC passes through the Latin-1 characters, i.e. does not convert them to Unicode. SearchServer's Translation Text Reader (nti), was chained on top of cTREC and the Win_1252_UCS2 translation was specified via its /t option to translate from Latin-1 to the Unicode character set desired by SearchServer.

3.3 Indexing

A separate SearchServer table was created for each language, created with a SearchSQL statement such as the following:

```
CREATE SCHEMA CLEF01DE CREATE TABLE CLEF01DE
(DOCNO VARCHAR(256) 128)
TABLE_LANGUAGE 'GERMAN'
STOPFILE 'GER_AW.STP'
PERIODIC
BASEPATH 'E:\DATA\CLEF';
```

The TABLE_LANGUAGE parameter specifies which language to use when performing linguistic operations at index time, such as breaking compound words into component words and stemming them to their base form. The STOPFILE parameter specifies a stop file containing typically a couple hundred stop words to not index; the stop file also contains instructions on changes to the default indexing rules, for example, to enable accent-indexing, or to change the apostrophe to a word separator. The PERIODIC parameter prevents immediate indexing of rows at insertion time. The BASEPATH parameter specifies the directory from which relative filenames of insert statements will be applied. The DOCNO column was assigned number 128 and a maximum length of 256 characters.

Here are the first few lines of the stop file used for the French task:

```
IAC = "\u0300-\u0345"
PST="'`"
STOPLIST =
a
à
afin
# 112 stop words not shown
```

The IAC line enables indexing of the specified accents (Unicode combining diacritical marks 0x0300-0x0345). Accent indexing was enabled for all runs except the Italian and English runs. Accents were known to be specified in the Italian queries but were not consistently used in the Italian documents. The PST line adds the specified characters (apostrophes in this case) to the list of word separators. The apostrophes were changed to word separators for all submitted runs except the German and English runs. Probably it would have made no difference to have also included it in the German runs. Note that the IAC syntax is new to SearchServer 5.0, and the interpretation of the PST line may differ from previous versions.

Into each table, we just needed to insert one row, specifying the top directory of the library files for the language, using an Insert statement such as the following:

```
INSERT INTO CLEF01DE ( FT_SFNAME, FT_FLIST ) VALUES
('GERMAN', 'cTREC/E/d=128:s!nti/t=Win_1252_UCS2:cTREC/C/@:s');
```

To index each table, we just executed a Validate Index statement such as the following:

```
VALIDATE INDEX CLEF01DE 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) (no CLEF collection actually required a sort file that big). The index includes a dictionary of the distinct words (after some Unicode-based normalizations, such as converting to upper-case and decomposed form) and a reference file with the locations of the word occurrences. Additionally, by default, each distinct word is stemmed and enough information saved so that SearchServer can efficiently find all occurrences of any word which has a particular stem.

4 Search Techniques

The CLEF organizers created 50 “topics” and translated them into many languages. Each translated topic set was provided in a separate file (e.g. the German topics were in a file called “Top-de01.txt”). The topics were numbered from C041 to C090. Each topic contained a “Title” (subject of the topic), “Description” (a one-sentence specification of the information need) and “Narrative” (more detailed guidelines for what a relevant document should or should not contain). The participants were asked to use the Title and Description fields for at least one automatic submission per task this year to facilitate comparison of results.

We created an ODBC application, called QueryToRankings.c, based on the example stsample.c program included with SearchServer, to parse the CLEF topics files, construct and execute corresponding SearchSQL queries, fetch the top 1000 rows, and write out the rows in the results format requested by CLEF. SELECT statements were issued with the SQLExecDirect api call. Fetches were done with SQLFetch (typically 1000 SQLFetch calls per query).

4.1 Intuitive Searching

For all runs, we used SearchServer's Intuitive Searching, i.e. the IS_ABOUT predicate of SearchSQL, which accepts unstructured text. For example, for the German version of topic C041, the Title was “Pestizide in Babykost” (Pesticides in Baby Food), and the Description was “Berichte über Pestizide in Babynahrung sind gesucht” (Find reports on pesticides in baby food). A corresponding SearchSQL query would be:

```
SELECT RELEVANCE('V2:3') AS REL, DOCNO
FROM CLEF01DE
WHERE FT_TEXT IS_ABOUT 'Pestizide in Babykost Berichte über Pestizide
in Babynahrung sind gesucht'
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. The statement "SET MAX_SEARCH_ROWS 1000" was previously executed so that the working table would contain at most 1000 rows.

4.2 Linguistic Expansion

SearchServer uses lexicon-based natural language processing technology to "stem" each distinct word to one or more base forms, called stems. For example, in English, "baby", "babied", "babies", "baby's" and "babying" all have "baby" as a stem. Compound words in languages such as German, Dutch and Finnish should produce multiple stems; e.g., in German, "babykost" has "baby" and "kost" as stems.

By default, Intuitive Searching stems each word in the query, counts the number of occurrences of each stem, and creates a vector. Optionally some stems are discarded (secondary term selection) if they have a high document frequency or to enforce a maximum number of stems, but we didn't discard any stems for our CLEF runs. The index is searched for documents containing terms which stem to any of the stems of the vector.

Whether or not to stem, whether or not to allow multiple stems per term, and which language to use are controlled by the VECTOR_GENERATOR set option:

Language	Recommended VECTOR_GENERATOR (SearchServer 5.0)
German	'word!ftelp/lang=german/base | * | word!ftelp/lang=german/expand'
French	'word!ftelp/lang=french/base/single | * | word!ftelp/lang=french/expand'
Italian	'word!ftelp/lang=italian/base/single | * | word!ftelp/lang=italian/expand'
Spanish	'word!ftelp/lang=spanish/base/single | * | word!ftelp/lang=spanish/expand'
Dutch	'word!ftelp/lang=dutch/base | * | word!ftelp/lang=dutch/expand'
English	'word!ftelp/lang=english/base/single | * | word!ftelp/lang=english/expand'

Table 2: Recommended VECTOR_GENERATOR Settings (SearchServer 5.0)

The above settings were the ones used for the submitted runs, and subsequent experiments (below) confirmed they were the best ones. The main issue was whether to specify "/base" or "/base/single" in the first part of the VECTOR_GENERATOR. Experiments on last year's collections found that "/base" worked better for German and "/base/single" worked better for French and Italian. For languages with compound words, such as German, "/base" is necessary for all components of the compound words to be included. For languages with few compound words, in the rare cases when multiple stems are returned, they often are spurious; e.g. in English, for "Acknowledgements", the 2 stems currently returned are "acknowledgement" and "acknowledgment", which appear to be just alternate spellings, so keeping both stems would improperly double-weight the term, hurting ranking. Based on this reasoning, we assumed "/base" would be better for Dutch (because Dutch contains many compound words), and "/base/single" would be better for Spanish (because it does not contain many compound words); again, experiments below confirmed these were the best choices.

Besides linguistic expansion, we did not do any other kinds of query expansion. For example, we did not use approximate text searching for spell-correction because the queries were believed to be spelled correctly. We did not use row expansion or any other kind of blind feedback technique.

4.3 Statistical Relevance Ranking

SearchServer calculates a relevance value for a row of a table with respect to a vector of stems based on several statistics. The inverse document frequency of the stem is estimated from information in the dictionary. The term frequency (number of occurrences of the stem in the row (including any term that stems to it)) is determined from the reference file. 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. The

already-mentioned count of the stem in the vector is also used. SearchServer synthesizes this information into a relevance value in a manner similar to [4] (particularly the Okapi approach to term frequency dampening) and also shares some elements of [5]. 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'). SearchServer's RELEVANCE_DLEN_IMP parameter controls the importance of document length (scale of 0 to 1000) to the ranking.

Because the evaluation program (trec_eval) may re-order rows with the same relevance value, we post-process the results files by adding a descending fraction (0.999, 0.998, 0.997, etc.) to the relevance values of the rows for each topic to ensure that the order of evaluation is identical to the order SearchServer returned the documents.

4.4 Query Stop Words

Our QueryToRankings program removed words such as "find", "relevant" and "document" from the topics before presenting them to SearchServer, i.e. words which are not stop words in general but were commonly used in the topics as general instructions. The lists for the CLEF languages were developed by examining the CLEF 2000 topics (not this year's topics). After receiving the relevance assessments this year, we did an experiment, and this step had only a minor benefit; the average precision increased just 1% or 2% in all languages (in absolute terms, from 0.0031 in Italian to 0.0107 in French). It doesn't appear to be important to comb the old topics files for potential query stop words.

5 Results

Below we present an analysis of our results, including results of some unofficial "diagnostic" runs. We look at the following evaluation measures: *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).

The Monolingual Information Retrieval tasks were to run 50 queries against document collections in the same language and submit a list of the top-1000 ranked documents to CLEF for judging (in June 2001). The 5 languages were German, French, Italian, Spanish and Dutch. CLEF produced a "qrels" file for each of the 5 tasks: a list of documents judged to be relevant or not relevant for each topic. From these, the evaluation measures were calculated with Chris Buckley's *trec_eval* program. Additionally, the CLEF organizers translated the topics into many more languages, including English, and also provided a comparable English document collection, for use in the multilingual task. By grepping the English results out of the multilingual qrels, we were able to produce a comparable monolingual English test collection for diagnostic runs.

For some topics and languages, no documents were judged relevant. The precision scores are just averaged over the number of topics for which at least one document was judged relevant.

When comparing two runs and finding that one run had a higher score on more topics than the other, we like to know how likely it is that this result could have happened by chance. For each result, we use the sign test to compute the *significance level* (also known as the p-value) of the result in relation to the hypothesis that when the two runs differ in their average precision score on a topic (by the 4th decimal place), they are equally likely to have the higher score. If the significance level is 1% or less, then we can say the hypothesis is contradicted by the result at the 1% level and consider the result to be statistically significant, i.e. this result is unlikely to have happened if the runs were really using equally viable approaches regarding the average precision measure.

The computation of the significance level is straightforward. Let f(x,n) = (n!/((n-x)!x!))*(0.5^n), which is the probability of x successes in n trials when the probability of success is 0.5. Let X_1 be the number of topics on which run 1 scored higher, and let X_2 be the number of topics on which run 2 scored higher. Let $n = X_1 + X_2$. Let $X_{min} = \min(X_1, X_2)$. If $X_1 <> X_2$, then the significance level is $2*(f(0,n) + f(1,n) + \ldots + f(X_{min},n))$. If $X_1 == X_2$, then the significance level is 100%. Example: if in 50 topics, one run scores higher in 34, lower in 15, and ties in 1, then the significance level is 2*(f(0,49) + f(1,49) + … + f(15,49)), which is 0.9%.

5.1 Submitted runs

Tables 3.1 to 3.5 show the precision scores of our submitted runs for each language of the Monolingual Information Retrieval task. The CLEF organizers have also given the participants the median average precision scores on each topic for each language. We show the number of topics on which our runs scored higher, lower and tied (to 4 decimal places) with the median in average precision, and compute the significance level.

For each language, at least one SearchServer run had more average precision scores above the median than below. The submitted German, Dutch and Spanish runs had a significance level of less than 1%.

All submitted runs used both the Title and Description fields. Runs humDE01, humFR01, humIT01, humES01 and humNL01 used relevance method 'V2:3' and RELEVANCE_DLEN_IMP 500; these settings typically worked best on last year's collections. Runs humDE01x, humFR01x, humIT01x, humES01x, humNL01x used relevance method 'V2:4' and RELEVANCE_DLEN_IMP 750, which worked well on the TREC-9 Main Web Task last year [6]. After receiving the relevance assessments, preliminary experiments suggest that a combination of 'V2:3' and 750 would have been best for most languages this year, but it makes little difference.

Run	AvgP	P@5	P@10	P@20	Rec0	Rec30	vs Median	SigL
humDE01	0.4403	54.7%	49.8%	44.6%	0.7836	0.5377	39-8-2	0.0%
humDE01x	0.4474	56.3%	51.4%	44.8%	0.8206	0.5521	36-9-4	0.0%
Median	0.3660	n/a	n/a	n/a	n/a	n/a	0-0-49	-

Table 3.1: Precision of Submitted German runs

Run	AvgP	P@5	P@10	P@20	Rec0	Rec30	vs Median	SigL
humFR01	0.4825	55.5%	44.3%	36.3%	0.8149	0.5936	28-14-7	4.4%
humFR01x	0.4789	49.8%	42.2%	35.4%	0.7788	0.5811	27-18-4	23%
Median	0.4635	n/a	n/a	n/a	n/a	n/a	0-0-49	-

Table 3.2: Precision of Submitted French runs

Run	AvgP	P@5	P@10	P@20	Rec0	Rec30	vs Median	SigL
humIT01	0.4555	54.0%	49.4%	41.2%	0.7830	0.5727	23-20-4	76%
humIT01x	0.4332	50.6%	46.2%	39.8%	0.7069	0.5421	15-30-2	2.6%
Median	0.4578	n/a	n/a	n/a	n/a	n/a	0-0-47	-

Table 3.3: Precision of Submitted Italian runs

Run	AvgP	P@5	P@10	P@20	Rec0	Rec30	vs Median	SigL
humES01	0.5378	68.6%	60.8%	52.8%	0.8708	0.6644	35-7-7	0.0%
humES01x	0.5363	68.2%	60.0%	51.1%	0.8514	0.6681	38-7-4	0.0%
Median	0.4976	n/a	n/a	n/a	n/a	n/a	0-0-49	-

Table 3.4: Precision of Submitted Spanish runs

Run	AvgP	P@5	P@10	P@20	Rec0	Rec30	vs Median	SigL
humNL01	0.3844	52.0%	41.8%	34.1%	0.7558	0.5145	39-6-5	0.0%
humNL01x	0.3831	50.4%	45.4%	33.7%	0.7504	0.4940	37-10-3	0.0%
Median	0.2986	n/a	n/a	n/a	n/a	n/a	0-0-50	-

Table 3.5: Precision of Submitted Dutch 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
vs Median: Number of topics on which the run scored higher, lower and equal (respectively) to the median average precision (to 4 decimal places)
SigL: Significance Level: the probability of a result at least as extreme (vs Median) assuming it is equally likely that a differing score will be higher or lower

5.2 Impact of Linguistic Expansion

To measure the benefits of SearchServer's linguistic technology, Tables 4.1 to 4.6 show runs which were done with a more recent SearchServer build in August 2001. For each language, the runs vary in their VECTOR_GENERATOR setting. The first run set VECTOR_GENERATOR to the empty string, which disables linguistic processing. The second run set /base but not /single in the first part of the VECTOR_GENERATOR, which allowed all stems for a term to be added to the vector. The third run set /base and /single in the first part of the VECTOR_GENERATOR, allowing only a single stem to be added to the vector for each term. For all these runs, both the Title and Description were used, the relevance method was 'V2:3' and the RELEVANCE_DLEN_IMP setting was 750:

Run	AvgP	P@5	P@10	P@20	Rec0	Rec30	vs no ling	SigL
DE: no ling	0.3269	46.1%	43.7%	36.4%	0.7102	0.4038	-	-
DE: /base	0.4669	58.0%	52.4%	47.4%	0.8186	0.5699	42-6-1	0.0%
DE: /base/single	0.3843	49.8%	45.7%	38.3%	0.7489	0.4957	35-13-1	0.2%

Table 4.1: Impact of Linguistic Expansion in German

Run	AvgP	P@5	P@10	P@20	Rec0	Rec30	vs no ling	SigL
FR: no ling	0.4114	46.9%	38.0%	33.0%	0.7301	0.5047	-	-
FR: /base	0.4712	49.8%	42.7%	35.5%	0.7991	0.5776	27-18-4	23%
FR: /base/single	0.4855	52.2%	43.1%	36.2%	0.8121	0.5927	28-17-4	14%

Table 4.2: Impact of Linguistic Expansion in French

Run	AvgP	P@5	P@10	P@20	Rec0	Rec30	vs no ling	SigL
IT: no ling	0.3900	48.1%	43.0%	35.5%	0.7076	0.4729	-	-
IT: /base	0.4309	50.6%	48.5%	40.1%	0.7401	0.5422	25-19-3	29%
IT: /base/single	0.4514	50.6%	48.5%	40.9%	0.7456	0.5553	28-16-3	9.6%

Table 4.3: Impact of Linguistic Expansion in Italian

Run	AvgP	P@5	P@10	P@20	Rec0	Rec30	vs no ling	SigL
ES: no ling	0.4848	63.3%	55.9%	47.1%	0.8117	0.6154	-	-
ES: /base	0.5316	68.2%	60.2%	51.0%	0.8621	0.6608	28-18-3	18%
ES: /base/single	0.5429	70.6%	61.6%	51.9%	0.8842	0.6726	33-13-3	0.5%

Table 4.4: Impact of Linguistic Expansion in Spanish

Run	AvgP	P@5	P@10	P@20	Rec0	Rec30	vs no ling	SigL
NL: no ling	0.3189	42.4%	36.8%	28.0%	0.6603	0.4197	-	-
NL: /base	0.4144	54.0%	45.6%	35.1%	0.7939	0.5313	37-10-3	0.0%
NL: /base/single	0.3700	49.2%	40.8%	32.6%	0.7349	0.4912	31-16-3	4.0%

Table 4.5: Impact of Linguistic Expansion in Dutch

Run	AvgP	P@5	P@10	P@20	Rec0	Rec30	vs no ling	SigL
EN: no ling	0.4732	48.9%	39.1%	29.7%	0.7765	0.5959		
EN: /base	0.5245	52.8%	41.5%	31.6%	0.8112	0.7026	26-13-8	5.3%
EN: /base/single	0.5317	52.8%	41.7%	31.7%	0.8213	0.7029	26-13-8	5.3%

Table 4.6: Impact of Linguistic Expansion in English

Impact of Linguistic Expansion: For all 6 investigated languages (German, French, Italian, Spanish, Dutch and English), SearchServer's linguistic expansion was found to boost all investigated precision scores, including average precision and Precision@5. For 3 languages, German, Dutch and Spanish, the results (of comparing the number of topics of higher and lower average precision with linguistics enabled and disabled) had a significance level of less than 1%.

Table 5 summarizes the percentage increases in the precision scores using the recommended VECTOR_GENERATOR setting compared to disabling linguistics. It appears that German and Dutch are the biggest beneficiaries of linguistic expansion. The percentage changes for French and Italian were generally larger than for Spanish, even though more topics were helped in Spanish:

Language	AvgP	P@5	P@10	P@20	Rec0	Rec30	Range
German	+43%	+26%	+20%	+30%	+15%	+41%	15-43%
Dutch	+30%	+27%	+24%	+25%	+20%	+27%	20-30%
French	+18%	+11%	+13%	+10%	+11%	+17%	10-18%
Italian	+16%	+5%	+13%	+15%	+5%	+17%	5-17%
Spanish	+12%	+12%	+10%	+10%	+9%	+9%	9-12%
English	+12%	+8%	+7%	+7%	+6%	+18%	6-18%

Table 5: Percentage Increase from Linguistic Expansion, Descending Order by Average Precision

References

[1] Cross-Language Evaluation Forum web site. http://www.clef-campaign.org/

[2] Martin Braschler. CLEF 2000 Result Overview. Slides of presentation at CLEF 2000 Workshop. http://www.iei.pi.cnr.it/DELOS/CLEF/CLEF_OVE.pdf

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

[4] 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

[5] 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

[6] Stephen Tomlinson and Tom Blackwell. Hummingbird's Fulcrum SearchServer at TREC-9. To appear in E.M. Voorhees and D.K. Harman, editors, *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*. NIST Special Publication 500-xxx. http://trec.nist.gov/pubs/trec9/t9_proceedings.html

---