TREC-5 Experiments at Dublin City University: Query Space Reduction, Spanish & Character Shape Encoding.

Fergus Kelledy & Alan F. Smeaton

{fkelledy, asmeaton}@compapp.dcu.ie

School of Computer Applications,
Dublin City University, Glasnevin,
Dublin 9, IRELAND

Abstract

In this paper we describe work done as part of the TREC-5 benchmarking exercise by a team from Dublin City University. In TREC-5 we had three activities as follows:

- Our ad hoc submissions employ Query Space Reduction techniques which attempt to minimise the amount of data processed by an IR search engine during the retrieval process. We submitted four runs for evaluation, two automatic and two manual with one automatic run and one manual run employing our Query Space Reduction techniques. The paper reports our findings in terms of retrieval effectiveness and also in terms of the savings we make in execution time.
- Our submission to the multi-lingual track (Spanish) in TREC-5 involves evaluating the performance of a new stemming algorithm for Spanish developed by Martin Porter. We submitted three runs for evaluation, two automatic, and one manual, involving a manual expansion from retrieved documents.
- Character shape coding (CSC) is a technique for representing scanned text using a much reduced alphabet. It has been developed by Larry Spitz of Daimler Benz as an alternative to full-scale OCR for paper documents. Some of our TREC-5 experiments have started evaluating the performance of a CSC representation of scanned documents for information retrieval and this paper outlines our future work in this area

Acknowledgements: The authors would like to acknowledge the contributions to the work reported here given by Martin Porter who developed and provided us with access to his Spanish stemming algorithm, and to Larry Spitz who developed the character shape coding technique and who also provided us with access to his code and his accumulation of knowledge and data in the area.

1. Introduction

The work reported in this overview is a description of the research we carried out for our efforts in TREC-5. In all we have submitted 13 official runs, DCU961 To DCU964 (4 runs) represent Category A ad hoc, DCU965 to DCU967 (3 runs) represent our work in Spanish and DCU968 to DCU96D (6 runs) represent our efforts in Character Shape Coding. These 13 submitted TREC-5 runs are based on three separate streams of research and are described independently in the following three sections of the paper.

For all our work, the platform we used for our experiments was a SUN SparcStation 5 with 64 Mbytes of RAM and 6 Gbytes of local disk space. All experiments detailed in this overview were carried out using our own IR search engine developed by us over the last two years.

2. Query Space Reduction

In this line of our research the issue of query response time is critical and is the motivational force for the work, as in the case with research carried out by [Bown95], [Pers95] and [Moff94]. It is, in our opinion, very important that IR systems return required information to a user in an acceptable amount of time and for us this is of equal importance as the effectiveness of an IR system being used. Traditional IR research has always concentrated on effectiveness and efficiency has been a poor relation. To this end we developed and implemented a number of Query Space modelling techniques which improve the efficiency of our experimental IR system.

Within our test environment we define the Query Space (QS) to be the amount of data from the inverted file which needs to be processed in order to satisfactorily respond to a users query. Figure 2.1 illustrates an abstract view of this data.

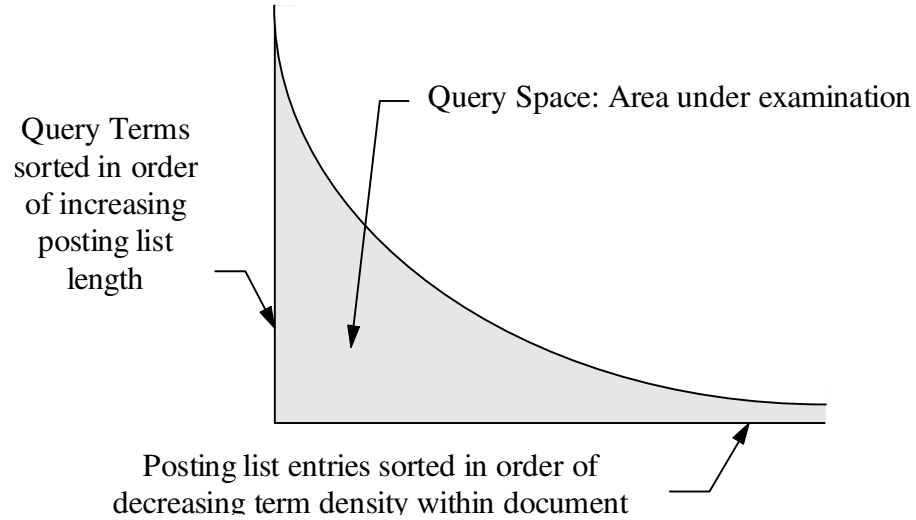

Figure 2.1 - Abstract View of Query Space.

The QS is composed of query terms and their corresponding posting lists. Query terms in the QS are those which occur both in the query and the document collection, i.e. their inclusion in the QS will have some impact on the final ranking of documents returned to the user in response to the query. For visualisation purposes the QS is best laid out in two dimensional space with the y-axis representing the query terms and the x-axis representing the posting lists of the query terms.

Within the QS, query terms are sorted in order of increasing posting list length. The posting lists themselves are ordered by decreasing within document term density. Due to the structure of the QS in which the posting lists are arranged vertically and horizontally in order of probable value to the query, one can assume that the concentration of relevant information with respect to the query is higher in the top left part of Figure 2.1 than in the bottom right part. This QS structure facilitates the notion of thresholding, in which not all of the QS need be processed thus greatly reducing I/O during retrieval.

2.1 Query Term Thresholding.

Query Term Thresholding (QTT) exploits the structure of the QS by removing the necessity to process all terms in the query. Figure 2.2 illustrates QTT as it applies to our QS, the shaded area represents the subsection of the QS that is actually processed during retrieval. This thresholding technique operates at the query term level i.e. based on the threshold value a query term is either included or excluded from the retrieval process. As one moves down the QS,

terms become more general (they occur in more documents) and therefore less useful in discriminating one document from another. These terms also occupy a significant percentage of the QS due to their frequency of occurrence within the collection.

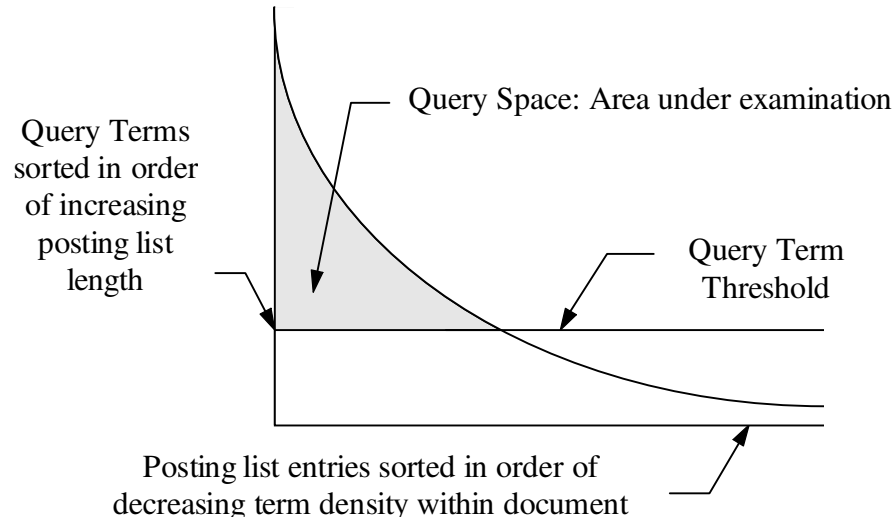

Figure 2.2 - Abstract View of Query Term Thresholding.

It therefore makes sense to attempt to reduce or eliminate the need to process these 'low value' query terms during the retrieval process. The incorporation of the QTT significantly reduces the processing and I/O costs of the retrieval process whilst also having a positive effect on the effectiveness of the retrieval process by eliminating 'noisy' postings from consideration during retrieval.

2.2 Posting List Thresholding.

As the posting list entries are sorted in order of decreasing 'within document term density', posting entries at the end of a posting list will be of less value in the retrieval process due to their low within-document term density. This fact presents the possibility of removing these 'low value' postings from consideration during the retrieval process. It makes sense that that more of the discriminating posting list entries (those with high IDF scores) entries should be processed and less of the non-discriminating posting list entries should be processed.

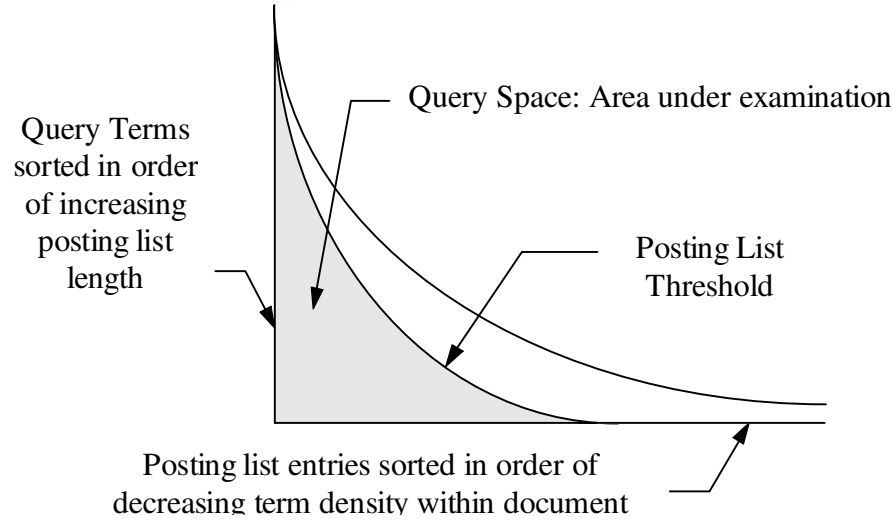

Figure 2.3 - Abstract View of Posting List Thresholding.

This results in a variable thresholding approach in which the PLT value is initially set to a high percentage of postings and is gradually lowered as each QS index term is processed. Figure 2.3 abstractly illustrates this thresholding process.

2.3 Query Term and Posting List Thresholding.

The QS thresholding techniques detailed in Sections 2.1 and 2.2 can operate in conjunction with each other on the same QS as illustrated in Figure 2.4 resulting in further reductions in the amount of the QS processed during retrieval.

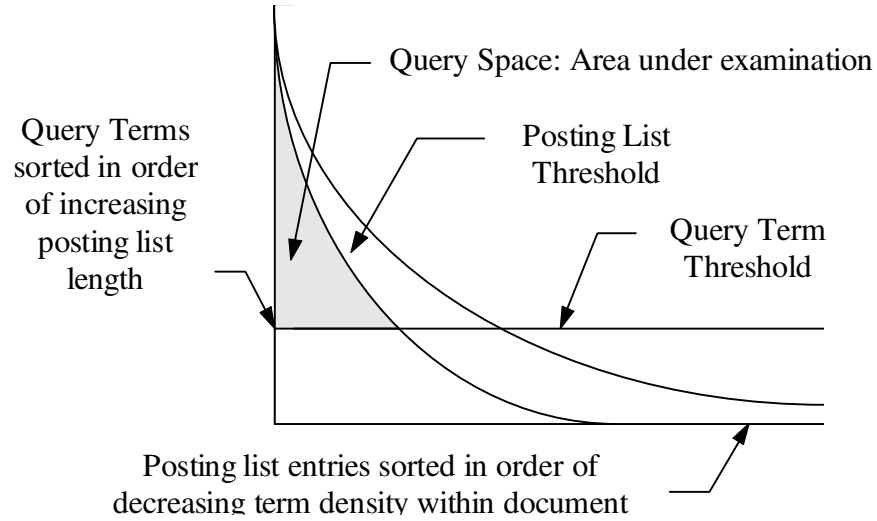

Figure 2.4 - Abstract View of Combined Thresholding Approach.

2.4 Document Accumulator Thresholding.

Document accumulators are used for summing up all of the partial query term similarity scores to form a final query-document similarity score. If no restriction is imposed on the activation of document accumulators then a large number of accumulators must be sorted in order to identify the *N* most highly-scored documents. Therefore some method for limiting the number of accumulators activated during retrieval is very important for efficiency.

The operation of our accumulator thresholding technique is as follows: new accumulators are created for all documents which achieve a non-zero query document similarity score until the maximum active accumulator limit is reached Once reached, only the already activated accumulators or registers are allowed sum more partial query document similarity scores. This places an upper limit on the number of accumulators activated hence reducing the load on the sort procedure which produces a final ranked list of documents.

2.5 Results of Query Space Reduction.

Figure 2.5 and Figure 2.6 detail the results obtained in terms of system effectiveness for our ad hoc submissions to TREC-5. We submitted two automatic runs (*DCU962, DCU964*) and two manual runs (*DCU961, DCU963*). *DCU961* and *DCU962* have the QSR techniques outlined earlier applied to them while *DCU963* and *DCU964* have no thresholding. The parameters chosen for these respective thresholding methods have been defined through experiments on the TREC-4 and TREC-3 collections. It can be clearly seen that there is no performance degradation in terms of system effectiveness caused by the inclusion of our QSR techniques. In fact the inclusion of our QSR techniques actually causes a slight improvement in

performance in terms of average precision for both the automatic (Avg. P moving from 0.1334 to 0.1340) and manual runs (Avg. P moving from 0.1804 to 0.1862).

	AUTOMATIC			**MANUAL**	
	DCU962 (QSR)	*DCU964 (No QSR)*		*DCU961 (QSR)*	*DCU963 (No QSR)*
P @ 0.0	0.4525	0.4404	P @ 0.0	0.5336	0.4952
P @ 0.1	0.2540	0.2531	P @ 0.1	0.3615	0.3507
P @ 0.2	0.2076	0.2067	P @ 0.2	0.2979	0.2897
P @ 0.3	0.1805	0.1802	P @ 0.3	0.2571	0.2492
P @ 0.4	0.1575	0.1571	P @ 0.4	0.2254	0.2171
P @ 0.5	0.1374	0.1380	P @ 0.5	0.1955	0.1883
P @ 0.6	0.1079	0.1085	P @ 0.6	0.1478	0.1437
P @ 0.7	0.0878	0.0843	P @ 0.7	0.1173	0.1136
P @ 0.8	0.0644	0.0639	P @ 0.8	0.0814	0.0756
P @ 0.9	0.0277	0.0275	P @ 0.9	0.0403	0.0367
P @ 1.0	0.0181	0.0192	P @ 1.0	0.0181	0.0177
Av. P	**0.1334**	**0.1340**	**Av. P**	**0.1862**	**0.1804**
P @ 10 docs	0.2540	0.2460	P @ 10 docs	0.3320	0.3160
P @ 30 docs	0.1867	0.1880	P @ 30 docs	0.2547	0.2427

Figure 2.5 - QSR Vs No QSR for Automatic and Manual runs.

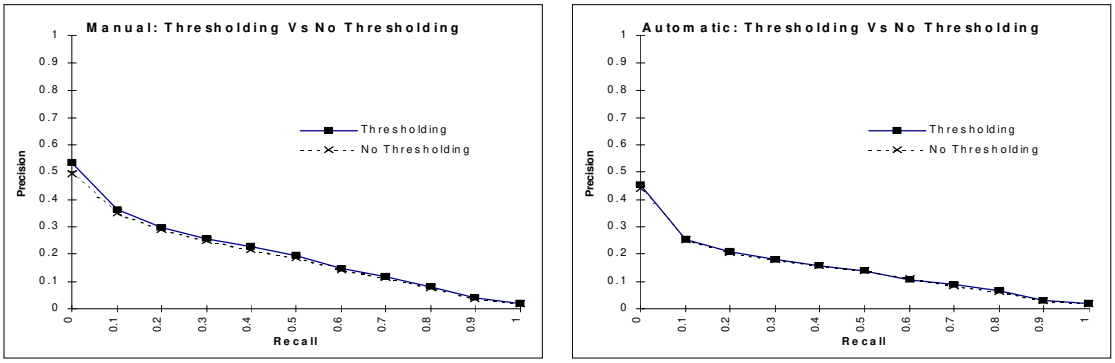

Figure 2.6 - Precision-Recall comparisons for Automatic & Manual runs.

In addition to measuring the effect of our QSR techniques on system effectiveness we also measured the effect on system efficiency in terms of CPU resources used during retrieval. For all runs the memory and processor loads were the same given the machine used was dedicated to this task.

	Automatic			**Manual**		
	Average	**Max**	**Min**	**Average**	**Max**	**Min**
No QSR	9.018	16.9	2.1	22.442	44.4	9.8
QSR	4.644	8.6	2.1	8.238	19.4	4.5
% Red.	48.50%			63.29%		

Figure 2.7 - Timing comparisons (in Seconds) for Automatic & Manual runs.

Figure 2.7 summarises the results we obtained for all of our ad hoc TREC-5 submissions. The average time spent processing a query for the automatic submission goes from 9.018 seconds to 4.644 seconds which represents a saving of 48.50% and the average time spent processing a query for the manual submission goes from 22.442 seconds to 8.238 seconds a

saving of 63.29%. The difference in savings obtained between the automatic and manual runs can be ascribed to the difference in the average size of the Query Spaces generated, with our system identifying on average 8.96 terms per query for the automatic run and an average of 30.9 terms per query for the manual run. The larger the QS generated the greater potential savings possible using our QSR techniques.

3. Spanish TREC-5

In TREC-3 and TREC-4, Dublin City University participated in the Spanish track. In TREC-3 we implemented a trigram matching procedure where trigrams were assigned IDF weights. In TREC-4 we used the NMSU Spanish POS tagger to index documents and texts by the base forms and POS tags for words and again used term weighting for retrieval.

Our involvement in the Spanish track for TREC-5 came out of a chance conversation with Martin Porter, developer of the much-used 1980 stemming algorithm for English. He now works for a company called MUSCAT who have built up a reputation for their products in the museum catalogue area and from that have developed a search engine for WWW sites called Euro*ferret*. Euro*ferret* is used by the Electronic Telegraph online newspaper and by UK government departments for their respective web pages. In an effort to expand the capabilities of Euro*ferret*, Martin Porter has developed a stemming algorithm for Spanish (as he has done for Italian and for French). The issue of Muscat LTD's intellectual property rights prevent a fuller description of the Spanish stemming algorithm and the details of its operation have not been published, but we include here a sketch of how it operates.

The basic plan for the Spanish stemmer is like that of the English stemming algorithm. Word endings are taken off if they come at the end of a stem of suitable length, as measured in syllables. As an ending is removed, the stemmer may attempt to remove further endings associated with it. Thus the ending -IVO or -IVA (like the English -IVE) has the associated ending -AT, and after taking off -IVO the stemmer looks for -AT, possibly taking off -ATIVO. There is a small graph structure for these endings which the implementation follows.

Porter's Spanish stemmer also has a short list of prefixes, which do not contribute to the evaluated length of the stem, for example CON-. If there is no significant morphological suffix of this type, the stemmer takes off a verb ending. This is one of the things that makes Porter's Spanish stemmer so different from and bigger than his stemmer for English. The romance languages all have highly inflected verb forms: seven tenses, each with six endings as well as imperative and participle forms, each having three different classes. Spanish and Italian also have forms in which direct and indirect object pronouns attach to the verb. All of these features are handled by the stemmer.

The stemmer takes special actions for irregular verbs. The English language has about 100 irregular verbs in use, e.g. DIG, DRAW, DRINK, DROVE, etc., but the irregularities are very simple (DIG/DUG, DRAW/DREW/DRAWN, etc.) and the irregular forms of verbs frequently mean different things, so any kind of normalisation is unwise (DUG can also mean a teat, DRUNK can also mean inebriated, DROVE can mean a large number of, etc.). In Spanish there are about 500 irregular verbs that Porter's stemmer knows about and their complexity suggests normalisation is advisable. To this list of irregular verbs one can add a batch of regular verbs with very short stems. The stemming algorithm then searches the whole word for a match with one of these verbs in one of its declinable forms and if found, can normalise it exactly.

For our involvement in TREC-5 Spanish we created our own list of Spanish stopwords, using a translation of Porter's stopword list for English supplemented with additional words from a Spanish grammar book for prepositions and other closed word categories. We removed

stopwords and stemmed the document and query texts in the TREC-5 collection. Using the search engine and term weighting functions developed for our mainline ad hoc experiments described earlier, we completed 3 official runs as follows:

- **DCU966**: we used query terms from the short form of the queries only (automatic); the average query length was 16.0 terms
- **DCU965**: we used the full form of the query only (manual); the average query length was 48.5 terms.
- **DCU967**: we used the short form of the query (run DCU966) to select the top 10 documents which were then presented to a Spanish speaker to read. Using these documents the user then selected extra terms thus constructing an expanded form of the query which, along with the original query terms, is used to re-rank the remainder of the unseen documents yielding an average query length of 59.2 terms.[1]
  Our results are reproduced in the table below.

	DCU966	**DCU965**	**DCU967**	**DCU967C**
P @ 0.0	0.6905	0.7661	0.8565	0.7149
P @ 0.1	0.6268	0.6335	0.6428	0.6171
P @ 0.2	0.4985	0.5383	0.5547	0.5147
P @ 0.3	0.4227	0.4719	0.4733	0.4629
P @ 0.4	0.3700	0.3985	0.4213	0.3976
P @ 0.5	0.3299	0.3404	0.3533	0.3327
P @ 0.6	0.2773	0.2819	0.2660	0.2630
P @ 0.7	0.2217	0.2213	0.1954	0.1935
P @ 0.8	0.1744	0.1697	0.1468	0.1458
P @ 0.9	0.1192	0.1020	0.0927	0.0923
P @ 1.0	0.0026	0.0021	0.0004	0.0004
Av. P	**0.3298**	**0.3482**	**0.3482**	**0.3258**
P @ 10 docs	0.4960	0.5760	0.5760	0.5000
P @ 30 docs	0.4507	0.4813	0.4820	0.4307

As would be expected, using the longer form of the queries (DCU965) yields better performance than using the shorter form (DCU966). Manually expanding the query from the initial short form (DCU967C) improves retrieval over simply using the short form at the high precision end of the scale, but not overall. Our manual expansion of queries by a Spanish speaker (native Irish) with no information retrieval background (DCU967C) does not yield as good retrieval as using the long form of the queries (DCU965).

In terms of comparison to other submitted results, the table below shows how our overall average precision compares with others' submitted runs.

	DCU966	DCU965	DCU967C
On or above Median	11	17	14
Below Median	14	8	11

[1] In our official submissions we erroneously submitted results for the complete collection re-ranking after query expansion and did not incorporate the top 10 rank positions of our overall ranking having been generated from the first pass. This has been corrected in the results coded as DCU967C and as the reader can see this has a big effect on results.

4. Character Shape Coding

Character shape codes (CSCs) are a reduced alphabet for representing the textual content of documents. In conventional optical character recognition, each token in a scanned image is mapped to a character in the full alphabet consisting of {A-Z, a-z, 0-9}. CSCs, developed by Larry Spitz, are an attempt to define a much-reduced alphabet based on the characteristics of character recognition from bitmaps of scanned documents. In essence CSCs are an attempt to group together characters based on the similarities of their printed characteristics. For example, in most printed fonts the letters "c" and "e" have similar characteristics in that neither rise above a horizontal line defined by the upper reaches of the letter "x", neither drop below the base line whereas the characters g, q, p, etc. do, and both "c" and "e" have an "eastword" concavity.

In previous work by Larry Spitz [Spit95a, Spit95b, Reyn95] he has defined a number of sets of CSCs using different alphabet mappings. A series of evaluations have also been carried out to examine the uniqueness of the CSC representations for the surface forms of words as they occur in text and these have illustrated a surprising uniqueness among CSC patterns. In our CSC experiments in TREC-5 we transformed the representation of document texts into 3 different CSC representations using the 3 different CSC mappings shown below.

CSC-V0

A	A-Z b d f h k l t
x	a c e m n o r s u v w x z
g	g p q y
i	I
j	j

CSC-V1

A	A-Z b d f h k l t
x	a m n o r s u v w x z
e	c e
g	g p q y
i	I
j	j

CSC-V2

A	A-Z b d f h k l t
x	a m o r s u v w x z
e	c e
g	g p q y
i	I
j	j
n	n

In effect what we are simulating is scanning and automatic transformation of paper documents into their CSC representations which is a much simpler and more accurate process than full-scale optical character representation and in our experiments we assume 100% accuracy in such CSC recognition. In later experiments we plan to introduce the naturally-occurring noise which such a process would have.

The effectiveness of a CSC representation for documents for information retrieval applications has not been evaluated before so our CSC experiments in TREC-5 we set out to do some preliminary work in this, using TREC to establish the groundwork for future experiments. In order to concentrate our work on the effectiveness of the CSC representations, we chose to do our experiments in category B ad hoc retrieval, using the 253 Mbyte text (74,520 documents) from the Wall Street Journal. In doing this we eliminated considerations of document length normalisation which have such a variance on experimental results as other

TREC groups have shown. The downside of doing this, however, is that we exclude ourselves from direct comparison with other groups taking part in the corruption track in TREC.

Having "indexed" documents by representing them as the CSC representations for all their surface word forms, we then processed queries in the following way. For both the manual (short form) and automatic (long form) queries we automatically removed stopwords and reverse-stemmed the non-stopwords. This was done by using Porter's stemming algorithm to stem a large portion (50 Mbytes) of WSJ texts and recording, for each unique word stem, the surface word form occurrence from the WSJ texts, of the word which yielded that stem. This pre-processing stage was used to generate our mapping of word stems to word occurrences. Each word stem in the query was then expanded into the set of surface forms of words in order to generate morphological (due to verb tense, noun plurals, etc.) and surface (due to various letter capitalisations) variants. The set of expanded terms per query was then pruned manually to eliminate errors due to stemming and unlikely word form occurrences. The remaining word form occurrences were turned into each of the three CSC representations and used as query terms for the three sets of experiments.

As an example, query 251 is :

"Exportation of Industry

Documents will report the exportation of some part of U.S. Industry to another country.

Relevant documents will identify the type of industry being exported, the country to which it is exported; and as well will reveal the number of jobs lost as a result of that exportation."

If we examine the term "*exportation*" as used in the query, it is stemmed to "export" which is the same stem as the following word forms which occur in our WSJ text *Export, export, Exports, Exports, exporting, exported*, etc. After manual pruning of the automatic generation of all possible word forms, the following is the list of query terms that was generated for q251:

type	types	lost	Export	Exporting	export	exported
exporter	exporters	country	countries	industry	industrial	industrialised
Industry	job	jobs	result	resulted	resulting	resultant

Each of these surface form occurrences was turned into a CSC token for each of the three mappings we use and the process described above yielded an average of 24.96 terms (CSC "words") for the short form of queries, and 29.4 terms for the long form of queries.

Each of these CSC terms in each mapping will map to multiple surface form occurrences in the document texts, and the distribution of these is very skewed. For example, the CSC-V1 code "exxgxxA" in query 251 above can map to the words *copout, expand* and *export* and in a comprehensive lexicon gathered by Lary Spitz these three surface word forms have been found. In our WSJ texts only the latter two are present. The CSC-V1 code "ixAxxAxixA" maps to the word *industrial* only both in Spitz's comprehensive lexicon and in our WSJ text. At the other extreme, the CSC-V1 code for the word *lost* is "AxxA" and shares that code with 194 other word forms in our WSJ text and 366 other words in Spitz's lexicon ! The table below shows, over the 50 short and long form queries in TREC-5, the average number of surface form occurrences which share the same CSC string, for each of the three CSC mappings we used, as measured in our WSJ text. It is to be expected that these numbers would be higher if

we had not used a domain-specific lexicon (our 50 Mbytes WSJ text) but a comprehensive lexicon instead.

	Short form of queries:	Long form of queries:
CSC-V0	24.88	23.72
CSC-V1	21.37	16.87
CSC-V2	13.30	10.41

This table confirms the expected results of the higher CSC mappings to word forms being more precise and less ambiguous and also shows a high degree of ambiguity with respect to surface form word occurrence, for the CSC tokens we use in queries. This is much higher than we would have expected

In our official TREC-5 category B submission there were 6 runs coded DCU968, -9, -A, -B, -C, and -D representing the use of short and long forms of queries for the three CSC mappings we used. Results are pretty poor and in the table below we show the total number of relevant documents found (from 1064 in total !) for each run:

Code	Meaning	Total reldocs Retrieved (45 topics)	Total reldocs Retrieved (36 topics)
DCU968	CSC-V0, short queries	21	21
DCU969	CSC-V0, long queries	21	20
DCU96A	CSC-V1, short queries	21	21
DCU96B	CSC-V1, long queries	20	19
DCU96C	CSC-V2, short queries	25	24
DCU96D	CSC-V2, long queries	17	16

The poor results above are easily explained by the noise introduced through the use of CSC tokens in queries which have such a high degree of surface form occurrences in text. Clearly the next set of experiments we should perform should involve manually or automatically selecting CSC terms for inclusion in queries based on the number of surface form occurrences sharing that CSC term. This is the subject of ongoing work..

5. Conclusions

The experiments reported in this paper as the Dublin City University submissions to TREC-5 represent mixed successes. The work on query space thresholding yields major improvements in retrieval efficiency without loss of retrieval effectiveness. Our experiments in Spanish track leave us about mid-table in terms of effectiveness compared to others' submissions though we have not incorporated any "smarts" such as phrase recognition, into the retrieval we used in our Spanish runs. The real question in our Spanish runs would be addressed by stemming the documents and texts with a different stemmer and measuring

retrieval effectiveness using the same retrieval environment (term weighting, etc.) we have used here. The final set of experiments on character shape coding must be acknowledged as being only preliminary work as is explained in the analysis of results and further work on CSC-based retrieval is presently ongoing.

6. References

[Bown95] "Fast Evaluation of Structured Queries for Information Retrieval", E. W. Brown, in: *Proc. SIGIR'95*, Seattle, 1995.

[Pers95] "Document Filtering for Fast Ranking", M. Persin, in: *Proc. SIGIR'94*, Dublin, 1994.

[Moff94] "Fast Ranking in Limited Space", A. Moffat, J. Zobel, in: *Proc. ICDE'94*, Houston, 1994.

[Reyn95] "Document Reconstruction; A Thousand Words from one Picture", J C Reynar et al, in: *Proc. Symposium on Document Analysis and Information Retrieval*, UNLV, 1995.

[Spit95a] "Using Character Shape Codes for Word Spotting in Document Images", A. L. Spitz, In *Shape, Structure and Pattern Recognition*, D Dori and A Bruckstein (Eds), World Scientific, Singapore, 1995.

[Spit95b] "An OCR Based on Character Shape Codes and Lexical Information", A L Spitz, in: *Proc. 3rd International Conference on Document Analysis and Recognition*, Montreal, Canada, 1995.