Indexing Structures Derived from Syntax in TREC-3: System Description

Alan F. Smeaton, Ruairi O'Donnell and Fergus Kelledy
{asmeaton, rodonnel, fkelledy}@CompApp.DCU.ie

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

Abstract: *This paper describes an approach to information retrieval based on a syntactic analysis of the document texts and user queries, and from that analysis, the construction of tree structures (TSAs) to encode and capture language ambiguities. TSAs are constructed at the clause level and thus each document can yield many TSAs and each query may be represented by several TSAs. The TSAs from documents and from queries are then matched and their degrees of overlap between individual TSAs are computed and then aggregated to yield a score for each document, which is then used in ranking the collection. This paper presents the system description when benchmarking our retrieval strategy on category B of TREC-3, i.e. on c.550 Mbytes of the Wall Street Journal newspaper texts. The implementation is based on a two-stage retrieval where a statistically-based pre-fetch retrieval retrieves the set of WSJ articles for the more computationally expensive language based processing. The results of our retrieval system in terms of precision and recall are disappointing and an analysis of why is also included. Part of this analysis includes a direct comparison between our system and some mainstream IR approaches. In addition to performing ad hoc retrieval on texts in English, we have also performed ad hoc retrieval on texts in Spanish using a weighted trigram approach, and this is outlined and performance results given in an appendix.*

1. Introduction and Motivation

Since 1989 the IR group at Dublin City University has been working on a method to index and retrieve texts based on a domain-independent syntactically-based language analysis. Our thesis has been that the meaning or content of a segment of natural language (query or document) can be at least partly represented by the words used, combined with the structures underlying the syntax used in the language. While it is true that syntax and syntactic structure does not always imply content, and it is equally true to say that there will be many instances of syntactic ambiguity after a syntactic analysis to make uncertain what true content is, our research has tried to address the question of whether there is enough content indication from syntax to overcome these obvious drawbacks to our approach.

In recent years, several other researchers have tried to use syntactic level analysis in the indexing and retrieval of texts. One approach taken has been to use syntactic structure and syntactic relationships between words to derive normalised phrases as indexing terms. Examples of this approach are the CLARIT work (Evans et al, 1993) and earlier work of Fagan (Fagan, 1987). Syntax has also been used to identify head-modifier relationships between words (Salton *et al*, 1990) or to identify dependent word pairs (Strzalkowski and Carbello, 1993). In all these cases, syntactic ambiguity occurring in language is either ignored or normalised. An alternative to this has been to incorporate syntactic ambiguities into the indexing and

retrieval operations and to use alternative interpretations of content, as indicated by syntactic structure, during retrieval. Alternative syntactic interpretations of language are usually best encoded in some kind of structured representation. The TINA project at Siemens, Münich (Schwarz, 1990) and the COP project at Pittsburgh (Metzler and Haas, 1989) are two such examples.

In published experimental evaluations of the effectiveness of document retrieval when using syntax in the ways mentioned above, the first approach, deriving normalised representations, has been the most successful. In fact large-scale evaluations of approaches which encode syntactic ambiguities have been non-existent to date. In our previously published work we have used syntactic analysis to build structures into which we have encoded syntactic ambiguities. These TSAs have been evaluated in phrase to phrase matching and the results we have obtained show promise for future work (Sheridan and Smeaton, 1992). Since then we have scaled phrase-phrase matching up to query-document matching and here we present our first results.

The remainder of this paper is organised as follows. In the next section we briefly outline the language analyser we have used in our work. Following that we present the structures we derive from syntax, known as TSAs and we describe how they are matched individually and how these matches are combined to compute document scores. In section 4 we describe our approach to implementing a document retrieval algorithm on the *WSJ* database. Section 5 presents the official TREC-3 results we obtained. As part of the analysis of these results we re-ran our retrieval strategy on 3 other document rankings obtained from 3 mainline IR research systems and these comparative results are also included here. Section 6 presents some general comments about the validity of our approach to document indexing and retrieval. Finally, in addition to performing retrieval on the *WSJ* we have also performed retrieval on Spanish texts and the approach we have taken here is outlined in an appendix.

2. ENGCG Analysis

The language analyser used in our work was developed at the Research Unit for Computational Linguistics at the University of Helsinki as part of the CEC-funded SIMPR project (1989-1992). It is known as ENGCG and it performs a domain-independent morphosyntactic analysis or tagging of running English text. It has a two-level lexicon and it based upon a constraint grammar.

ENGCG works in 5 stages as follows. The first stage is the assignment of clause and sentence boundaries. This is followed by lexical and morphological analysis where each word is decomposed into all its possible morphological base forms, suffixes and prefixes. The lexicon has c.65,000 entries which recognise over 300,000 word forms. The lexicon also contains about 600 collocations or multi-word terms. As a result of this analysis, many words will have more than one tag so the next stage is a context-sensitive disambiguation to discard all and only the contextually illegitimate morphological readings. This is done using a constraint grammar with c.1,100 rules. In the next stage of the analysis, all possible syntactic functions are assigned to each word token and these tokens describe how a word affects and is affected by other words around it. These syntactic function labels are preceded by a '@' character and modifiers are denoted by a '>' or '<' depending on the direction of the modification. For example, the label '@AN>' indicates that the current word is an adjective (A) modifying a noun (N) somewhere to its right (>). In the final stage of the analysis, the syntactic function labels are disambiguated as much as possible by removing all and only the contextually illegal readings, according to a second constraint grammar of c.300 rules.

In the version we used here, 3% to 6% of words in general texts would be assigned more than one

tag after analysis but the *WSJ* texts contain many proper nouns so perhaps this figure may be higher for this data. By default a word not in the lexicon is treated as a proper noun. There are 32 possible function labels in the analysis output which we group into 6 categories namely heads, modifiers, verbs, adverbs, adjective and stopwords. Examples of stopwords would be determiners, auxiliaries, conjunctions, quantifiers, negation, etc.

In order to give some sense of the output of the ENGCG analyser, a sample of 1 Mbyte of *WSJ* text was analysed. This consisted of 153,198 words or 19,423 clauses, which were assigned an average of 1.138 syntactic labels per word. The occurrence of labels per category is given at the end of this paper and illustrates the kind of distribution to be expected.

Category	*Freq.*	*% Freq*
Modifier	47,098	27.0
Stopword	34,915	20.0
Head	37,590	21.6
Adverb	22,712	13.0
Adjective	9,731	5.6
Verb	22,331	12.8
Total	174,377	100.0

Table 2.1 Distribution of Label Occurrences in sample 1 Mbytes *WSJ* text

Further analysis of this sample text was then performed to determine which types of ambiguity are actually identified in the ENGCG output. We computed the occurrences of words assigned labels from more than one of our 6 categories and found that in the sample of 153,198 words, 7,344 are identified as either a modifier or a head, or put another way, of the 37,590 head category labels assigned, nearly 20% can also be modifiers. Analysis showed that this is due to the subject or object of a sentence also acting as a prepositional compliment. Of the 22,712 adverbs, 6,153 (27%) can also be modifiers which is due to occurrences of phrasal verbs.

In terms of syntactic rather than lexical ambiguities, we counted the number of instances of prepositional phrase attachment ambiguities and found that among the 19,423 clauses there were 4,569 instances. We also looked for ambiguity in complex noun phrases containing a conjunction whose role is ambiguous, i.e. conjoining modifiers or heads ("bearing cups and cones" being @NN> @OBJ @CC @OBJ) and we could find only 806 such occurrences, a surprisingly small figure. Even more surprising was the only 180 instances of conjunction among modifiers where the first modifier could also be a head ("hub and bearing components" being {@NN>,@OBJ} @CC @NN> @OBJ).

An example analysis of some running text, taken from WSJ870324-001 is given here for the sentence "Industry sources put the value of the proposed acquisition at more than 100 million.":

Industry *[industry] @NN>*
sources *[source] @subj*
put *[put] @+FMAINV*
the *[the] @DN>*
value *[value] @OBJ*
of *[of] @<NOM-OF*
the *[the] @DN>*
proposed *[propose] @AN>*
acquisition *[acquisition] @<P*
at *[at] @NOM @ADV*
more=than *[more=than] @AD-A>*
100 *[100] @qn>*
million *[million] @<P*

The example above shows, for each word in the input sentence, the base form in [] brackets and the syntactic label(s) assigned. The full ENGCG analysis also includes much morphological information on each word which is not shown above. We can also see multi-word lexical entries (more=than) and syntactic ambiguity in this sample.

One of the things which hindered our work was the processing speed of the parser we used which is an early deliverable from the SIMPR project and which runs at 8 to 10 words per second.

Lingsoft Inc have developed the speed but not the performance of this parser and taken it to market with a claimed processing speed of 500 words per second on a SUN SparcStation 10 model 30. The version we have is unsupported and we had no funds to upgrade or buy in support. In addition to speed, we also found some other problems with the analyser like the fact that it cannot handle calendar expressions as it does not use a calendar expression subgrammar, a plural noun cannot be a noun premodifier (industrial **relations** council ?), many ellipses are missed, an adjective cannot be a prepositional compliment (the bartender in my **father's** local ?), conjoined premodifier separated by commas are not correctly assigned. These kind of aberrations would be eliminated in the more recent version of the parser.

3. TSA Construction and Matching

In our original paper describing our TSAs (Sheridan and Smeaton, 1992) we generated one TSA per clause. These TSAs were quite complex in structure and hence in manipulations on them. We have since decided to reduce the complexity of TSA structures in the light of our findings re frequency of occurrence of different types of ambiguity. In the present format, each clause generates a list where each item in the list is either a verb, adverb, adjective, stopword or TSA for a noun phrase. For example, the full representation for the sentence given earlier is shown below and the representation for an entire document text would be a set of such lists, one per sentence.

In computing a score between a document and a query there are a large number of factors to be accounted for. Given that a document will be represented by a set of lists, one list per clause, and elements of these lists will be tree-structures, we must consider scoring a query-document match at the following levels:

- Leaf node level, covering matched base forms, matching between identical or related syntactic function labels

- Structural level, covering scores for inexact matches caused by ambiguities and combining node scores into one overall TSA score for clause matches, possibly normalising to account for clause length

- Document level, addressing how clause TSA scores are combined to produce document scores, possibly normalising by clause count.

We have made many experimental runs using TREC-2 queries and relevance assessments and the algorithm described here has been found to give best performance, though there is no guarantee it is optimal.

Overall, the document scoring algorithm works by matching each document TSA against each query TSA and adding the score for the best match to a running total. The next document TSA is then matched against all query TSAs and the score for the best-matched is added to the total and so on until all document TSAs have been processed. This approach presupposes that fragments of information content as represented in clauses, can be repeated in document texts but not in queries. In TREC queries a sub-topic of information need can be repeated within different fields of the query. For example in topic 101 the description field asks for "... *configuration, components and technology of the U.S.'s 'star wars' anti-missile defense system*" while the narrative field asks for "... *design and technology to be used in the anti-missile defense system advocated by ..., also known as 'star wars.'*". Effectively this same sub-topic is repeated in the narrative field but in a more descriptive manner. Our assumption is that where a sub-topic or clause appears in a document text it should be matched against one sub-topic of the query, the one it scores highest against. In aggregating TSA scores for a document we are thus treating TSAs as atomic

units but we have no notion of re-occurrence of TSAs in queries as they are too short. The same topic or TSA representation of that topic can re-occur and be re-scored where it occurs in a document text, where there are on average approximately 60 TSAs. The total score for the document is then normalised by dividing by the number of clauses (hence TSAs) in the document.

Computing the TSA score between two clauses is rather complicated and we will illustrate part of its operation by working through a match between a query clause **"lexical analysis within IR"** and a document clause **"effective IR systems which involve lexical and syntactic analysis of text"**. The TSAs for these are given in Appendix B at the end of the paper, in an abbreviated form. The leaf nodes also hold morphological information on words and the numbers in parentheses beside each word are labels to illustrate the sequence of the matching algorithm. Asterisk nodes illustrate a modification relationship with left child modifying right.

The TSA match starts with node 1 of the query and we find this is a modification relationship so we descend to node 3 and search the document TSA for a base form match which we find at node 12. We now compute a word-word match between these nodes. The document frequency of the baseform *analysis* is 3136 documents so its IDF weight is 2.713 and this is further weighted by a factor of 8 giving 21.9. This factoring comes from a set of rules determining that an adverb (@O-ADVL) and an object (@OBJ) are both significant labels in their respective clauses. The full set of such rules and weights comes from research for the TSA phrase to phrase matching.

We then return to query node 1 and descend to its left child, node 2 whose baseform is matched against node 10 in the document. The baseform frequency of the word *lexical* in *WSJ* texts is only 3 documents, so the IDF weight is 5.762, again weighted by a factor of 8 giving a word-word score of 46.1 as both are playing the same role (modifiers) within their respective clauses. The common ancestors of the matching nodes in respective TSAs are * (node 1) for the query and * (node 8) for the document, indicating both have the same modification relationships but in the residual structure between nodes 10 and 12 of the text TSA we find a conjunction among modifiers which is an indicator of ambiguity so the cumulative score to date (21.9+46.1=68.0) is further incremented by a bonus of only 5 (again from another rule set).

Moving on to query node 4 we find it has a right leaf child (5) and we search for *ir* in the document TSA and find it matches node 4. Its document frequency is 4 and its IDF weight is again multiplied by a factor of 8, the factor assigned to a match between a prepositional compliment (@<P, effectively the head of a prepositional phrase) and a premodifying adjective (@AN>) giving an additional weight of 45.1. The query term *within* does not occur in the document TSA so there is no contribution to the score there. Possible reductions in the overall degree of match caused by syntactic ambiguity due to propositional phrase attachment or ambiguities due to conjunctions in complex noun phrases, for example, would then be applied but they do not affect the occurrences in the present document TSA.

Finally, in addition to normalising query-document TSA scores by a within-document TSA count as described earlier, we also normalise TSA-TSA scores by dividing the word-word and syntactic structure scores by the number of nodes in the query TSA, 5 in this case, giving a final score of 23.6.

This worked example illustrates some inadequacies with our approach. For example, the term IR is an acronym but these are not handled in our retrieval. The most obvious problem with the approach, however, is its complexity which will obviously affect the implementation as described in the next section. A more subtle consequence of the complexity for retrieval effectiveness might be that the way we retrieve now is simply too complex and has lost

sight of the original motivation so the contribution of "structure from syntax implying content" may be too diluted in the overall retrieval.

4. Implementation of Document Retrieval

Because of the computational overhead of calculating query-document similarities using TSA-based matching we took a 2-stage approach to implementing retrieval. For each TREC query we used all fields given and we analysed the text using the ENGCG analyser. Words assigned only one of the syntactic function labels in our stopword category are discarded as are words whose baseforms occur in more than 10% of the documents in the collection. For the remaining words we calculate their IDF weight using their baseforms as indexing terms and we score documents using tf*IDF weighting on these baseforms. The top 1000 ranked documents are then used as input into the second stage of the retrieval process, TSA-based weighting. In earlier experiments we varied the size of the document set returned by this pre-search but we found that quantities above 1000 did not affect subsequent retrieval to any noticeable degree.

The tf*IDF weighting on baseforms which we use as a pre-search is implemented by using an inverted file. storage of the TSA structures was a bit more problematic however. We looked at using an OODBMS to store the TSA structures internally but found that we would have to decompose them into linear lists with extra fields as pointers and the storage overheads would then become too much of a burden. In the end we decided not to store TSA structures internally within GemStone but to use GemStone to manage TSA locations within documents. TSAs are in fact reconstructed on demand from an encoded representation of the ENGCG analysis, a process which is reasonably fast and which allowed us flexibility during our experimental runs if we decided to modify the TSA format.

A second consequence of the computational and storage overhead of our implementation is that we would have been unable to process the entire TREC-3 collection of 2 Gbytes so we elected to become category B participants. Our retrieval was implemented on 550 Mbytes of *Wall Street Journal* newspaper stories from 1986 to 1992, made up of 173,256 documents and c.27,000,000 words. Processing was done on a SUN SparcServer 690MP.

5. Experimental Results

The Dublin City University team submitted two official runs to be evaluated as part of TREC-3 ad hoc retrieval. These were given the codes DCUNL1 and DCUNL2 and are repeated from the proceedings below.

Recall	**Precision** *(DCUNL1)*	**Precision** *(DCUNL2)*
0.00	0.6431	0.5453
0.10	0.3973	0.3362
0.20	0.3383	0.2587
0.30	0.2931	0.2193
0.40	0.2321	0.1799
0.50	0.1907	0.1367
0.60	0.1479	0.1039
0.70	0.0999	0.0651
0.80	0.0610	0.0425
0.90	0.0299	0.0212
1.00	0.0097	0.0061
Avg Precision	0.1966	0.1514
P at 30 docs	0.2933	0.2433

Table 5.1 : Comparison of Results for DCUNL1 and DCUNL2

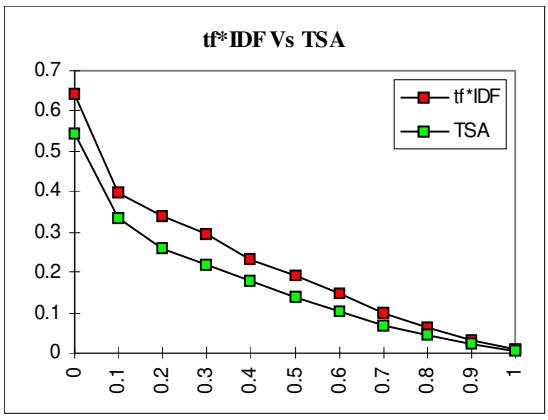


Figure 5.1 : Comparison of Results for DCUNL1 and DCUNL2

In comparison to results obtained by other TREC-3 groups, both category B and A, our results are disappointing. The most disappointing aspect of our results, however, is that our baseline prefetch run (DCUNL1) is actually **better** than our TSA-based run (DCUNL2). This is really quite unexpected. In experimental runs with TREC-2 queries we found that the TSA-based re-ranking improved retrieval slightly at the high precision end of the scale but averaged out as approximately equal to the tf*IDF weighting. In essence what we were finding in our pre-official runs was that our pre-fetch retrieval (tf*IDF weighting on ENGCG word baseforms) was not bringing in as many relevant documents as we wanted and our TSA-based retrieval was improving precision slightly and only at the high precision levels. In an attempt to determine if this was a fault of the quality of the pre-fetch we used, we contacted some other TREC-3 groups to see how our TSA-based retrieval would perform on top of other pre-fetch methods. One difficulty was that there are few category B groups in TREC-3 anyway so after discussion so we elected to use results provided to us using the following systems which are very heterogeneous in nature and hence should retrieve different document sets:

- Chris Buckley at Cornell sent us results he computed for TREC-2 topics (101-150) on category B data using the SMART system with official TREC-2 parameter settings. These results were used to provide a benchmark for TREC-2 category B participants and appeared in the TREC-2 proceedings (pp10).

- Sue Dumais at Bellcore did a special run of LSI-based retrieval on category B data and TREC-2 topics (101-150) where singular valued decomposition was used to reduce the term space. The parameter settings used here were determined from the full category A dataset.

- Jaime Callan at UMass provided results of a run with the INQUERY system on category A data but with only the category B *WSJ* articles extracted from that ranking. The parameter settings here were from the official UMass TREC-2 submission.

In each of these three cases we ran our TSA-based retrieval on the top 1000 document rankings and the results we obtained are presented below:

Recall	Precision	Precision
	Cornell	*TSA*
0.00,	0.8076	0.5736
0.10,	0.6292	0.4347
0.20,	0.5455	0.3461
0.30,	0.4901	0.2821
0.40,	0.4424	0.2478
0.50,	0.3654	0.2097
0.60,	0.3056	0.1788
0.70,	0.2497	0.1405
0.80,	0.1814	0.1052
0.90,	0.0986	0.0577
1.00,	0.0165	0.0051

Table 5.2 : Comparison of Results for TSA and Cornells

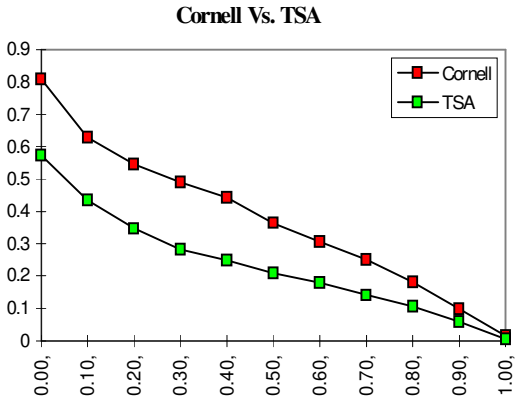


Figure 5.2 : Comparison of Results for TSA and Cornells

Recall	Precision	Precision
	LSI	*TSA*
0	0.7004	0.5721
0.1	0.5124	0.3777
0.2	0.4112	0.3073
0.3	0.3428	0.2612
0.4	0.3003	0.2219
0.5	0.2528	0.1717
0.6	0.2123	0.1319
0.7	0.1672	0.0917
0.8	0.1242	0.0642
0.9	0.0735	0.0339
1	0.0062	0.0043

Table 5.3 : Comparison of Results for TSA and LSI

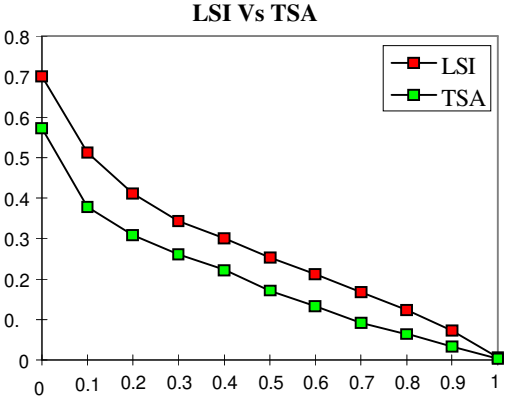


Figure 5.3 : Comparison of Results for TSA and LSI

	Precision	Precision
	INQ	*TSA*
0	0.8002	0.6066
0.1	0.6736	0.4487
0.2	0.5747	0.38
0.3	0.5042	0.3335
0.4	0.4565	0.3017
0.5	0.3984	0.2715
0.6	0.3463	0.2309
0.7	0.2652	0.1659
0.8	0.171	0.0828
0.9	0.0579	0.0195
1	0.0245	0.0087

Table 5.4 : Comparison of Results for TSA and Inquery

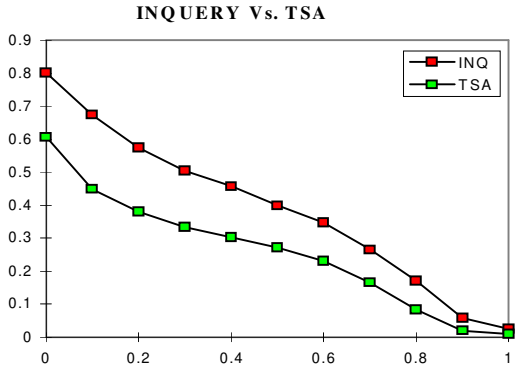


Figure 5.4 : Comparison of Results for TSA and Cornells

As can be seen from the results presented above, we have consistently and significantly managed to disimprove the results of pre-fetch runs from other IR approaches. These results conclusively confirm that the approach of using syntax to determine structural relationships between words and to use them in the way we have described as part of an information retrieval strategy, does not work.

6. Conclusions

There are a variety of reasons why our results are so poor, some of which we postulate here. In the

first place it seems that we score highly, those documents which have query terms in a syntactic relationship in documents which is similar to the relationship in the query, however results suggest that documents are relevant to queries for some other reason besides sharing structural syntactic relationships. A second reason for our poor performance could be that the version of the ENGCG language analyser we used is of poor quality; certainly it is slow ! After experimentation, we found that the score decrement due to syntactic ambiguity is quite low which suggests that this ambiguity does not really matter to retrieval, yet it is a mainstay of our approach. Finally, it has been suggested that the type of language used in TREC topic descriptions is very different to that used in document texts, interrogative vs descriptive language, and this suggests that the two language types should be treated differently, something which we did not do.

As a result of discussions at TREC it has been suggested that instead of piggy-backing TSA-based weighting **on top of** tf*IDF weighting, we could have had better results had we developed a TSA-based retrieval strategy which was totally independent of any kind of term weighting. It has been shown by a number of groups in TREC-3 and elsewhere (UMass, ETH, etc) that combining the results of more than one **independent** retrieval strategy into one overall global document ranking can bootstrap the performance of the individual retrieval strategies. Our TSA-based retrieval could have retrieved documents not retrieved by the term weighting strategy and thus could have been usefully combined, though this could be said of any approach to indexing and retrieval which is independent of any others.

The real contribution of our work, we feel, is the fact that we have at last tried to use structural aspects of syntactic relationships, as part of retrieval. The fact that we have failed to improve retrieval performance and have in fact considerably disimproved retrieval performance tells us that this line of research is not worth pursuing further.

Appendix A: Spanish TREC-3

In addition to taking part in ad hoc retrieval for English texts, the DCU team also took part in ad hoc retrieval from Spanish texts using a weighted trigram approach. Trigram-based retrieval has been tried in TREC-2 by Cavnar (Cavnar, 1993) in both adhoc and routing and the results which have been obtained are poor by TREC-2 standards, though Cavnar believes much better results are obtainable using this approach. After some variations and cul-de-sacs, the following are the steps taken in our official, submitted retrieval runs:

- After ftp-ing and decompressing Spanish, punctuation and numeric characters are discarded and the remaining text is converted to lowercase. Letters in the Spanish character set include upper and lowercase versions of the 26 used in English (a-z, A-Z) plus upper and lowercase versions of *ñ, á, é, í, ó, ú* and the double letter combinations *ll, ch* which can be treated as single characters. These extra characters beyond the 26 a to z are represented in our system by uppercase A to Z equivalents (N, A, E, I, O, and U). Other groups running retrieval on Spanish texts discarded all accents which, given the inconsistent use of accents in our TREC-3 Mexican newspaper texts, may have been the best approach. Such inconsistent use of accents is not so common in Spanish Spanish texts

- The stoplist used in Porter's stemming algorithm was translated into Spanish. These include multiword stopwords which were then treated as several single stopwords. A frequency count of words from a sample of 10 Mbtes of the text was generated and presented to our Spanish colleagues and they added further entries to our stoplist yielding a total of 245 entries. Documents were then stripped of these stopwords and this reduced the size of the documents by c.35%.

- Trigrams, 3 letter overlapping substrings, are then generated from the remaining words which are padded with one space before and after each word. For example, the word sueño (meaning dream) generates the trigrams _su, sue, ueN, eNo, No_, where "_" denotes a space and N denotes ñ.

  In analysing the text, after stopword removal, we found a total of 15,861 unique trigram occurrences. The 20 most frequently occurring of these were as follows:

res	_co	ent	nte	ado
_de	_re	est	iOn	_pr
sta	_in	_es	_ca	_se
con	ciO	aci	_di	ica

  What is interesting about this is that the most frequently occurring trigram (res) occurred 57,605 times while the 20th most frequently occurring (ica) occurred 54,856 times, so there was a very "flat" and non-Zipfian distribution among the most frequently occurring terms. This does not auger well for a retrieval strategy based on highly weighting the more discriminating terms, or trigrams in our case.

- The same process is then applied to Spanish queries. In our official run we used the topic description and narrative fields and we did not do any processing to remove any noise text.

- An inverted file of trigram occurrences is generated (which is 160 Mbytes in size) to speed up subsequent processing and to determine the frequency of occurrence of trigrams in the document collection.

- A score is calculated for each document which is based on *tf*IDF* weighting as

follows. For each trigram generated from the query, the document frequency was calculated and query trigrams occurring in more than α% of the documents were discarded. The remainder were sorted by increasing document frequency and an IDF weight calculated for each. Document scores were computed by processing query trigrams in order of increasing document frequency (lower IDF weight). Postings lists per trigram had previously been sorted by decreasing within-document frequency and as the algorithm worked through query trigrams, successively less of the postings lists were used in computing document scores thus effectively eliminating processing those with lower within-document frequencies. Finally, in computing document scores we kept accumulators for only a fixed number of documents, the first μ encountered.

- Documents are then ranked by these scores and the top 1000 per query are returned for evaluation.

Our system is implemented on a SUN SparcServer and returns a document ranking in 10-15 seconds for a TREC-3 topic description. In our official submission we chose parameter settings for α and μ (50% and 6800 respectively) which worked best for an implementation on English texts indexed by word stems rather than Spanish texts indexed by trigrams. This was a consequence of having no known queries or relevance assessments with which to work.

After the TREC-3 results and relevance assessments became known, we re-ran our experiments a number of times varying α from 5% to 50% in increments of 5% and for each of these varying μ from 1000 to 7000 in increments of 1000. At this point these would have been termed unjudged runs in that the retrieved documents may or may not have been judged by relevance assessors but nonetheless we obtained significant improvements, a consequence of our bad starting points, not the quality of our subsequent retrieval performance. The average precision values for each run show the best parameter settings for Spanish TREC-3 are 20% and 4000 accumulators. What is more important is that we have improved our official results considerably as the table below shows:

Recall	**Precision** *Official run*	**Precision** *Post-Official run*
0.00	0.5483	0.5279
0.10	0.1050	0.3590
0.20	0.0509	0.2608
0.30	0.0449	0.2187
0.40	0.0388	0.1768
0.50	0.0256	0.1362
0.60	0.0080	0.1082
0.70	0.0032	0.0853
0.80	0.0032	0.0565
0.90	0	0.0317
1.00	0	0
Avg Precision	0.0411	0.1612

Table : Results for Spanish TREC Official and Post-Official

The conclusion from our work on Spanish TREC-3 is that our performance in terms of the performances of other groups taking part in Spanish TREC-3 are worse than using weighted quadgrams which are worse than using any kind of crude stemming algorithm and term weighting strategies. This suggests that even though the character set may be larger than for English, trigram based retrieval may not be best. Certainly, trigram based retrieval would probably be good at handling word-word variants as would be generated from OCR and spelling errors but for straightforward retrieval, alternative approaches seem more promising. A more complete description of our work on Spanish TREC-3 is available via ftp (email one of the authors for details).

Acknowledgements: We would like to acknowledge the support of the Commission of

the European Communities under the VALUE program, project CCM-448, "Re-Implementation of SIMPR TSAs" and Dublin City University, in developing this work. We are also grateful to Sue Dumais (Bellcore), Chris Buckley (Cornell) and Jamie Callan (UMass) for either performing runs especially for us, or for providing us with some of their earlier results, and to Bill Richardson and Ciara Walsh from the School of Applied Languages at DCU for helping us with Spanish TREC.

Bibliography

(Cavnar, 1993) "N-Gram-Based Text Filtering for TREC-2", W. Cavnar, *in: The Second Text REtrieval Conference (TREC-2), D.K. Harman (Ed), NIST Special Publication 500-215,* 1993.

(Evans et al, 1993) "Design and Evaluation of the CLARIT-TREC-2 System", D Evans and R Lefferts, *in: The Second Text REtrieval Conference (TREC-2), D.K. Harman (Ed), NIST Special Publication 500-215,* 1993.

(Fagan, 1987) "Experiments in Automatic Phrase Indexing for Document Retrieval: A Comparison of Syntactic and Non-Syntactic Methods", J.L. Fagan, Technical Report 87-868, (Ph.D. Thesis), Department of Computer Science, Cornell University, September 1987.

(Metzler and Haas, 1989) "The Constituent Object Parser: Syntactic Structure Matching for Information Retrieval", D.P Metzler and S.W. Haas, *ACM Transactions on Information Systems,* **7**(3), 292-316, 1989.

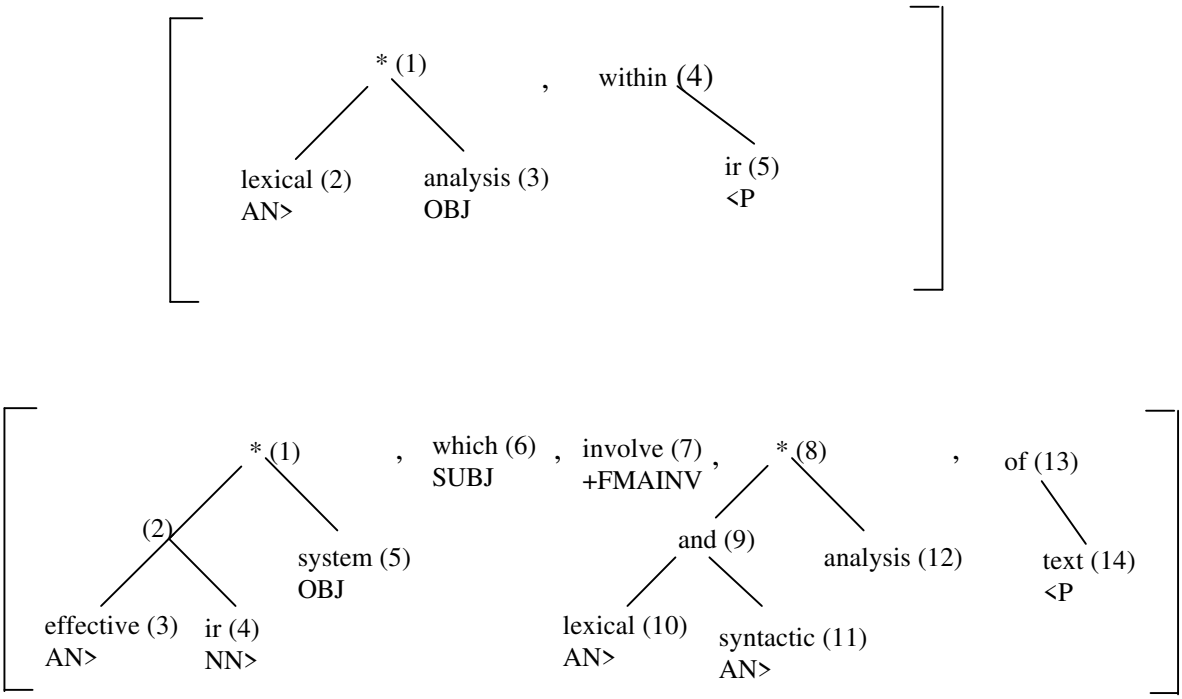
(Salton *et al*, 1990) "On the Application of Syntactic Methodologies in Automatic Text Analysis", G. Salton et al., *Information Processing and Management,* **26** (1), 73-92, 1990.

(Schwarz, 1990) "Content Based text Handling", C. Schwarz, *Information Processing and Management,* **26** (2), 219-226, 1990.

(Sheridan and Smeaton, 1992) "The Application of Morpho-syntactic Language Processing to Effective Phrase Matching", P. Sheridan and A.F. Smeaton, *Information Processing and Management,* **28** (2), 1992.

(Strzalkowski and Carbello, 1993) "Recent Developments in Natural Language Text Retrieval", T. Strzalkowski and J Carbello, *in: The Second Text REtrieval Conference (TREC-2), D.K. Harman (Ed), NIST Special Publication 500-215,* 1993.

Appendix B:Sample TSAs



TSAs for sample query and document clauses