EXPERIMENTS IN TERABYTE SEARCHING, GENOMIC RETRIEVAL AND NOVELTY DETECTION FOR TREC-2004.

Stephen Blott[2], Oisin Boydell[3], Fabrice Camous[2,] Paul Ferguson[1], Georgina Gaughan[1], Cathal Gurrin[1], Gareth J. F. Jones[1], Noel Murphy[1], Noel O'Connor[1], Alan F. Smeaton[1, 4], Barry Smyth[3], Peter Wilkins[1]

[1]*Center for Digital Video Processing, Dublin City University, Glasnevin, Dublin 9, IRELAND*

[2]*School of Computing, Dublin City University, Glasnevin, Dublin 9, IRELAND*

[3]*Smart Media Institute, Dept. of Computer Science, University College Dublin, Belfield, Dublin 4, IRELAND*

[4] Contact author: alan.smeaton@computing.dcu.ie)

Abstract: In TREC2004, Dublin City University took part in three tracks, Terabyte (in collaboration with University College Dublin), Genomic and Novelty. In this paper we will discuss each track separately and present separate conclusions from this work. In addition, we present a general description of a text retrieval engine that we have developed in the last year to support our experiments into large scale, distributed information retrieval, which underlies all of the track experiments described in this document.

1. INTRODUCTION

In TREC2004, Dublin City University took part in three tracks, Terabyte, Genomic and Novelty. In this paper section 2 presents a general description of a text retrieval engine that supports all experiments and sections 3-5 discuss each track separately and present separate conclusions for each track. Our experiments in Terabyte searching primarily focused on the integration of document structure and anchor text evidence to support enhanced retrieval performance. For the Genomics track we experimented with combining MEDLINE abstract searching with MEDLINE MeSH (Medical Subject Headings controlled thesaurus) field matching and for Novelty we evaluated a number of techniques for identifying novel information by exploiting various term characteristics though adaptation of traditional IR approaches.

2. GENERAL SEARCH ENGINE DESCRIPTION

This section will detail in general terms the workings of the underlying text retrieval engine (referred to as *Físréal*) that we developed within the past year in the Center for Digital Video Processing (CDVP). Since Físréal has been used in all of our TREC-2004 experiments we briefly discuss its operation and architecture. Section 2.1 will discuss the indexing process and section 2.2 will cover the retrieval engine architecture and associated retrieval concepts.

Físréal was primarily developed to provide fast search capabilities over very large amounts of data (i.e. of Terabyte size or greater), whilst at the same time providing a framework that would allow for experimental retrieval approaches to be quickly deployed and observed. A second requirement of Físréal was that it should be flexible enough to allow non-envisioned search tasks or the searching heterogeneous data types to be achieved.

As with any type of search system, the first necessary component is a representation of the search candidate data in a form to allow for retrieval, in this case an index and indexer.

2.1 Indexing

Físréal, as noted previously, was utilised by experiments for all three tracks in which we participated. Each of the data sets for these tracks was comprised of text, but due to differences between the source data, indexing required different preprocessing techniques for data from each track. However whilst the method of indexing varied from case to case, the end result in each instance was an index that was structured similarly to all other indexes created. The structure of the index we employed relies closely on the underlying filesystem and is to be documented elsewhere.

2.2 Retrieval

The following section will detail at a high level the architecture that was employed by our search engine, and introduce the concept of 'pipelines' which allow us to rapidly configure different retrieval methods.

2.2.1 Retrieval Engine Architecture

The retrieval engine was designed primarily to act as a distributed search engine made up of a series of 'leaf' search engines (or nodes) which would be invoked by an 'aggregate' search engine. An aggregate search engine is the same as any other instance of the search engine (leaf node) except that it handles all incoming search requests. Any given installation of a retrieval engine could integrate as many leaf nodes as required, or could operate as a single leaf node in a standalone fashion, see *Figure 1*:

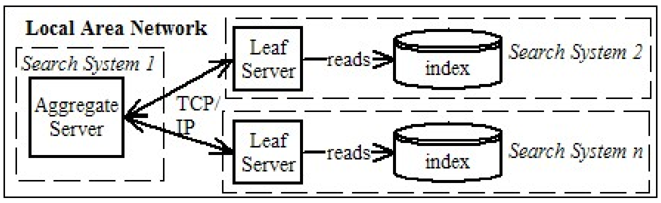

Figure 1: General Architecture of Físréal

The setup as defined above is the setup used for the Terabyte track (with four leaf nodes[1]). For each of the other tracks however only one leaf node was required. Each leaf server is in itself a totally independent instance of a search engine. Figure 2 provides a high-level architecture overview of each search engine instance:

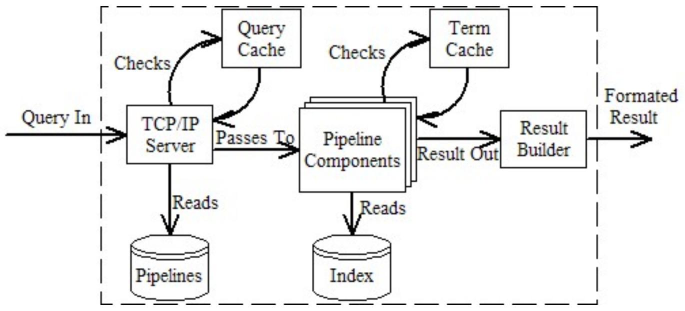

Figure 2: High-level architecture of a leaf node

As can be seen above, the search engines utilize a two-tier cache, a query cache followed by a term cache. The first tier or query cache examines incoming queries and if a match is found in the query cache then the result can be directly returned without further processing. The second or term cache is employed for any terms not picked up by the query cache and serves to further reduce the number of disk accesses. It is important to note that the caches were turned off for each track. The diagram also introduces the 'Pipelines' file. For any given search engine the retrieval methods that it can employ are defined by its associated pipeline file. Each incoming query to a search engine defines which pipeline is to process its query. We now briefly discuss pipelines.

[1] The documents were distributed to the four leaf nodes without any examination of their contents for optimisation purposes.

2.2.2 Pipelines

A pipeline is comprised of a series of pipeline components, self contained blocks of code that implement a common interface. Each pipeline component has only two methods it needs to implement, namely StartUp and Process. StartUp is invoked at search engine start time and any loading of files or setup actions occur within this method. The Process method is the only entry point for a pipeline component within an executing search engine. The argument and return value for this method is a Search Data Unit or SDU. An SDU is an augmented container object that amongst other attributes defines the following; the initial query, the pipeline to call, the result formatting type and an empty container for results. A pipeline component will take in a SDU, modify or add to it, then pass it onto the next pipeline component. The following diagram provides a definition of a basic BM25 pipeline with pipeline components illustrated.

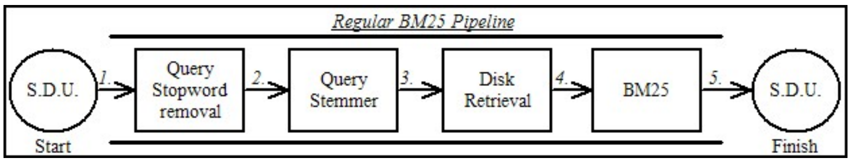

Figure 3: A Basic BM25 Pipeline Component

In Step 1, a SDU has been generated by the server for the standard BM25 pipeline. The first step of this pipeline is query stopword removal. In step 2, the SDU is moved onto the next component which will stem the stopped query within the SDU before (in Step 3) the relevant index entries are placed into the SDU. Ranking of these entries occurs utilising BM25 (in Step 4). This is the end of the pipeline, the SDU exits having had its query modified and result space filled. The above example pipeline would be represented in the Pipelines.xml file by the following entry:

```
<Pipeline name="BM25Pipeline">
   <component id="ie.dcu.cdvp.search.index.QueryStopWordRemovalComponent">
   </component>
   <component id="ie.dcu.cdvp.search.index.QueryStemmerComponent">
   </component>
   <component id="ie.dcu.cdvp.search.index.RetrivalPipelineComponent">
      <Accessor type="ie.dcu.cdvp.search.index.TermFileAccessor"/>
   </component>
   <component id="ie.dcu.cdvp.search.index.BM25PipelineComponent">
   </component>
</Pipeline>
```

New pipelines can thus be formed by creating a new entry in a Pipelines.xml configuration file and aligning the various components into the desired order. This made the reconfiguration of Físreál for different TREC tasks, quite straightforward.

3. TERABYTE TRACK ACTIVITIES

The TREC2004 Terabyte task used the new .GOV2 collection and required the processing of fifty topics (in standard TREC format). All of our runs were automatic, with the title of the topic being used as the query for each topic, with no manual intervention.

Our experiments can be divided into two distinct groups, and will now be presented accordingly. Three of our five runs were based on utilising either document text, URL text or anchor text. The last two runs, though based on our baseline run, integrated collaborative search techniques in order to generate improved ranked lists of results. We begin by discussing our three runs that are based on utilising the document text, URL text and the anchor text for retrieval and refer to these as our Content Experiments.

3.1 Content-Experiments

Our content experiments consisted of three runs, one a baseline run and two additional runs that utilise additional sources of evidence (URL text and anchor text) from the collection.

3.1.1 Baseline Run

For our baseline content-only run we used Okapi BM25 (Robertson et al., 2004) to rank the results, with k1=1.2, k3=1000 and b=0.75. Físréal supports sorted index creation, thus allowing us to sort document instances for terms in descending order of document normalized TF values. In this way we could read from disk only a subset of the top ranked entries for each term in order to improve retrieval latency. Given that the query time was an important factor of the experiments, we choose to read the top ten thousand documents for each term at retrieval time. This figure of ten thousand was chosen after comparisons between the result sets where all the documents were read versus a subset of the top documents, as well as taking the query timings into consideration to try and find a good balance between precision and speed.

For this run we returned the top 10,000 document Ids and refer to this run as DcuTB04Base.

3.2 Weighted BM25

The purpose of our second run was to examine what benefit we could gain by utilizing HTML markup elements to weight certain terms more than others. In Table 1 we outline what we considered to be the tags that would contain the most representative words of each document. Then for these tags we defined the extra weighting that we felt any word contained in these tag deserved. Normal body text was weighted at 1.0.

Table 1, Tag weights employed for DcuTB04Wbm25

Tag	TITLE	B	H1	H2	H3	H4	H5	H6	I	EM	U	A	ALT
Weight	6	4	4	4	2	2	2	2	2	2	2	4	6

Again Okapi BM25 is used with similar parameters to rank the top ten thousand documents for each term and this run was labeled DcuTB04Wbm25.

3.3 BM25, URL & Anchor Text Run

It is believed that integrating anchor text into the retrieval process can be useful in improving retrieval performance for Web IR (Craswell and Hawking, 2003). We generated anchor text surrogate documents by extracting the anchor text (with a window of 56 bytes either side) from all documents that link to a given document. We created an index from these documents and used Okapi BM25 ranking again here, but with parameter values k1=50, k3=1000, and b=0. These parameters are best guesses and will be the subject of further experimentation.

One additional source of evidence we incorporated into the ranking process for this run was the URL text of each document, where the document content was based on extracting terms from the URL string utilizing URL syntax to identify partitioning characters. This provided an additional index which was queried using Okapi BM25 with more conventional parameter weighting of k1 = 1.2, k3 = 1000 and b = 0.75.

The combination parameters for combining scores from each source of evidence used was as follows (text=0.6, anchor=0.3, URL=0.15). We were interested to see if these parameters (our best guess) could improve retrieval performance over a baseline of our Weighted BM25 run. This run is labeled DcuTB04Combo**.**

3.4 Collaborative Search Experiments

Our other two runs for the TREC Terabyte search track were based on the idea of collaborative search, which we briefly describe in the following section.

3.4.1 Background

The I-SPY Web search engine (Freyne et al., 2004) developed at University College Dublin proposes an approach to search known as collaborative search. The basic intuition is that the world of web search can be both repetitive and regular. Like-minded users tend to use similar queries and select similar results in response to these queries. Collaborative search takes advantage of this by reusing the past search behavior of similar users within well-defined communities of like-minded individuals. For example, an I-SPY search box located on a motoring Web site will attract motoring related queries and result selections so that ambiguous queries such as "jaguar" will result in the selection of car-related sites rather than those relating to cats or operating systems of the same name.

To achieve this, I-SPY operates as a meta-search engine (See Figure 4), dispatching queries to multiple traditional (underlying) search engines and merging their result-lists. However, I-SPY also tracks the results that are selected for given queries and stores this information in a so-called hit-matrix. The importance of the hit-matrix is that it allows I-SPY to estimate the relevance of a page p_i for a query q_j in terms of the relative proportion of times that p_i has been selected for those queries that are similar to q_j. Crucially, this relevance information can be used directly as a means to promote and rank those results that have been previously selected for q_j or related queries. When I-SPY receives a new target query q_T, in addition to retrieving a set of results from its underlying search engines, it also locates a set of results from its hit-matrix, by locating all queries that are above a set similarity threshold with the target query as in Formula 1.

$$WRel(p_i, q_T, q_1, ..., q_n) = \frac{\sum_{j=1...n} \text{Relevance}(p_i, q_j) \bullet \mathit{Similarity}(q_T, q_j)}{\sum_{i=1...n} \text{Exists}(p_i, q_j) \bullet \mathit{Similarity}(q_T, q_j)} \qquad (1)$$

These hit-matrix results are ranked according to their weighted relevance metric, and then combined with those from the underlying search engines.

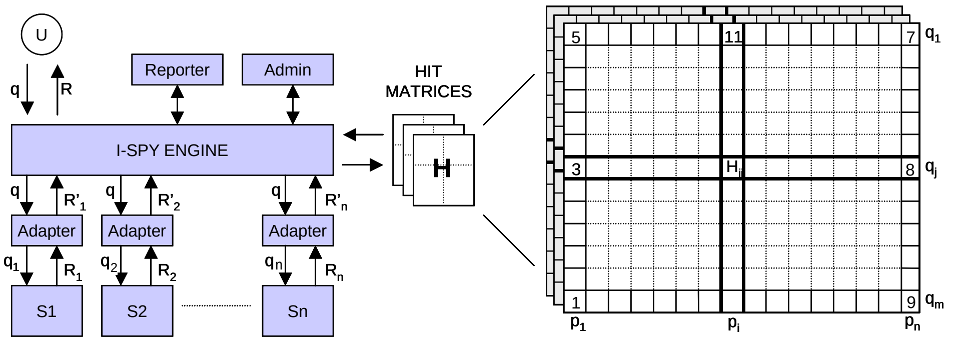

Figure 4: The I-SPY architecture and hit-matrix

A separate hit-matrix is maintained for each well-defined community of users, and in this way I-SPY adapts its ranking of results to reflect the past preferences of a given community by promoting those results that have been preferred in the past.

3.4.2 Applying I-SPY to the Terabyte Track

It is worth noting that I-SPY's approach is not only limited to the use of user selection information as a source of relevance. In fact, we can view the hit-matrix as a general mechanism for implementing a wide range of relevance feedback strategies that are based on any number of factors. To this end we apply I-SPY to two different Terabyte search engines, both of which are instantiations of the Físréal search engine developed at CDVP, DCU. The first is the search engine that is based on the standard Terabyte track document index as discussed in Section 1. The second is a similar search engine, but one that is based on the anchor-text index (as described in section 3.3). Both search engines were configured to use BM25 as the ranking algorithm.

To apply I-SPY to these document collections we must first train its hit-matrix to reflect our relevance model of choice, which involves the following steps:

1) A set of training queries must be generated. In this work we derive these queries by selecting sets of terms from the narrative and description of each topic.
2) These queries are submitted to I-SPY and the returned results are processed for entry into the hit-matrix according to a suitable selection model. In Section 3.1.3 we describe the use of a simple selection model based on the inverse rank of the top 100 returned results.

3.4.3 The I-SPY Terabyte Track Runs

We produced two separate runs of I-SPY which differ in terms of the underlying search engines that I-SPY calls upon and in the manner in which hit-matrix training occurs. For the first I-SPY run (DcuTB04Ucd1) we configured I-SPY to operate with both the standard document index and the anchor-text index as underlying search engines. We generated 500 training queries per topic with each query being between 2 and 8 terms in length. Each query was generated by combining words from the topic's narrative and description sections after stop-word removal. During training, each query was submitted to I-SPY and the top 100 results returned were used to update the hit-matrix using a linear inverse ranking function. The query-result hit-matrix entry for the top result is incremented by 100, with the increment for each subsequent result reduced by a constant amount.

The second I-SPY run (DcuTB04Ucd2) is similar to the first except that it relied on the anchor-text index alone as its underlying search engine, only 250 queries per topic were used in training, and the hit-matrix was updated with only the top 20 results per query. When training was completed the system was ready to accept queries.

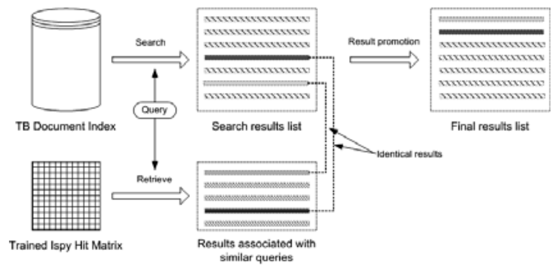

Figure 5: Generating the final result list

When a new query is submitted to a trained version of I-SPY, it needs to combine the results obtained from its underlying search engines (described earlier in section 3) with those results obtained from the hit-matrix. For both DcuTB04Ucd1 and DcuTB04Ucd2, we used the search engine (Físréal) that is based on the standard Terabyte track document index as the single underlying search engine when submitting the test queries. Each query was used to probe the hit-matrix, retrieving the entries associated with all similar queries (above a set similarity threshold with the original query) and ranking these results according to the weighted relevance metric outlined above. In DcuTB04Ucd1 we set the similarity threshold at ≥50% and in DcuTB04Ucd2 it was set at >0%; the former reused queries that shared 50% or more of their terms with the test query while the latter reused queries that shared at least 1 term with the test query.

In both of our TREC Terabyte runs we actively promoted the hit-matrix results, which also appear in the top 10,000 results returned from the underlying search engines, ahead of the results from the underlying search engines, and ranked them by their weighted relevance, as shown in Figure 5. This promotion is intended to improve the search precision. The promoted results are from related searches using more descriptive terms from the topic's narrative and description sections, and also from searches on the anchor text index.

4. GENOMIC TRACK ACTIVITIES

This section describes the methods used to respond to the challenge of the TrecGen 2004 ad hoc retrieval task. It follows up last year participation to the primary task (ad hoc) of TrecGen 2003 (Blott et al., 2003). For this previous participation we had used pseudo-relevance feedback, also known as "retrieval feedback", in order to improve a baseline provided by Jacques Savoy of the University of Neuchatel, Switzerland. For TrecGen 2004, however, we used Físréal to provide our own underlying search facility. Físréal allowed us to build different indexes for document representation (section 4.3.1) and use appropriate algorithms to retrieve and rank documents. The TrecGen 2004 ad hoc task involved dealing with new types of information needs and a much bigger document collection compared to the previous year, a subset of MEDLINE consisting of 10 years of completed citations inclusive from 1994 to 2003 and the total number of documents was 4,591,008. The 50 topics were obtained through interviews with biologists and thus reflect authentic needs. Each topic included a title (abbreviated statement of the information need), an information need (full statement of the information need) and a context (background information to put the information need in context). Five sample topics were provided with partial relevance judgements relative to these topics. This section is organized in the following way: in section 4.1, we present the main specificity of this year approach and briefly introduce the MEDLINE document record format (http://www.ncbi.nlm.nih.gov/entrez/query/static/help/pmhelp.html#MEDLINEDisplayFormat) and the Medical Subject Heading controlled vocabulary (http://www.nlm.nih.gov/mesh/meshhome.html). In section 4.2 we discuss our retrieval strategy which includes our indexing method, query construction and ranking strategies before presenting our results in section 4.3.

4.1 The use of MeSH descriptors

MEDLINE records have several fields including title, abstract, author and MeSH fields (http://www.ncbi.nlm.nih.gov/entrez/query/static/help/pmhelp.html#MEDLINEDisplayFormat). Some fields are not mandatory and they may not appear in all MEDLINE records. However, very few documents do not have any MeSH fields in the TrecGen 2004 collection. MeSH is the National Library of Medicine's controlled vocabulary thesaurus (http://www.nlm.nih.gov/). MeSH descriptors are manually added to the MEDLINE records by experts after the reviewing the full text of the article. We believe that the use of descriptors in the query should bring some consistency to the retrieval results as similar articles will have similar or identical descriptors, whereas the vocabulary used in their titles and abstracts might differ due to synonymy and polysemy of the natural and scientific languages. There are 22,568 unique descriptors available in the current version of MeSH, and the average number of descriptors per document in the TrecGen 2004 collection is 12. The MeSH thesaurus is organized hierarchically and includes 15 trees where MeSH descriptors or "Main Headings" (MH) may appear several times (1.8 times on average). In our TrecGen 2004 experiment, we chose to represent the documents as "bags of descriptors". We did not use the trees of the hierarchy to compute a tree similarity measures between documents and queries.

4.1.1 Our approach

As we mentioned above, the terms used in the title and abstract fields of MEDLINE records vary from one author to an author and they will represent a subset of the terms used in the full article. However, we have seen that MeSH terms are added to the MEDLINE record by a human expert after reading the full article. Also MeSH terms are part of a controlled vocabulary and will therefore be more consistent from one expert reviewer to another. A study by Funk and Reid (1983) showed that the indexing of MEDLINE records was highly consistent and represented "the state of the art in manually indexed data bases". Accordingly we decided to have a dual representation for each record: a "bag of title/abstract terms" and a "bag of descriptors" which allowed us to build two queries for each topic; one was a bag of terms extracted automatically from the different fields of the topic and the other one was a bag of descriptors that was created manually by a human expert after reading the topic fields content. The second query can be seen as an expansion of the first. The process of document representation and query building is described in detail in sections 4.3.1 and 4.3.2. We want to show that integrating two representations of MEDLINE records, a thesaurus-based one and a "free" text one, can improve the retrieval performance.

4.1.2 Previous and related work

Using the MeSH vocabulary for query expansion has been studied before. Srinivasan (1996) experimented with query expansion and retrieval feedback using an inter-field statistical thesaurus with a collection of 2,334 documents with 75 user queries. The thesaurus was built by looking at co-occurrences in documents of terms from the title/abstract fields and terms from the MH field. The co-occurrences with the best scores were used to expand the user queries and build MeSH queries that were used to search the MH fields of the documents. Using Cornell University's SMART retrieval system, Srinivasan combined different SMART retrieval strategies with different thesaurus-building strategies. The improvements obtained over the different baselines by the thesaurus expansion strategies were in the range of 8-10.6%. This approach contrasts with the experiments described here, in that we did not here build an inter-field thesaurus, but rather, asked an expert to analyse the 50 TrecGen 2004 queries and search the MeSH vocabulary with the MeSH browser available on the Web. We intend to build our own statistical inter-field thesaurus in the future and compare its expansion queries to the ones build by our expert.

4.2 Retrieval Strategy

We will discuss our retrieval strategy in terms of indexing, query construction and ranking strategies.

4.2.1 Indexing Methodology

We built two distinct indices for the TrecGen 2004 document collection: a title/abstract index and a MeSH index. For the title/abstract index, terms were extracted from the title (TI) and abstract (AB) fields of the MEDLINE records. All characters were made lowercase and stopwords taken from the MEDLINE stopword list (http://www.ncbi.nlm.nih.gov/entrez/query/static/help/pmhelp.html#Stopwords) were removed. No stemming was performed.

For the MeSH index we extracted all the descriptor from the Main Heading (MH) fields of the MEDLINE records. A star is used in MEDLINE records (e.g. "*Carbohydrate Sequence") to differentiate a "central concept" from another descriptor. A central concept is a descriptor that describes the central topic of the article whereas a non-central descriptor would describe peripheral issues also mentioned in the article. The descriptors are often accompanied by a "qualifier" or "subheading" that allows the domain of application for the chosen descriptor to be specified. In the MH field of the MEDLINE record, the subheading is added directly after the Main Heading it qualifies, with a forward slash preceding it. For example, "Immunoglobulin G/chemistry" is a Main Heading (MH)/subheading (SH) pair. The notion of "central concept" can also apply to a combination of MH and SH. In that case, a star is added at the beginning of the qualifier (e.g. "Immunoglobulin G/*metabolism"). Although we intend to use these distinctions in future work, in TrecGen 2004 we limited our experiments to represent documents as "bags of descriptors" and did not include the subheadings. So we removed all subheadings and made no distinctions between the "starred" Main Headings and other Main Headings. The end result of this was that we had two indices where documents were represented as bags of terms.

4.2.2 Query construction

For the title/abstract query construction (referred to as "ta" queries below), we chose two strategies. The first consisted in only taking terms from the title and the information need fields of the topic (we refer to this as the "tn" strategy). The second was to extract terms from each field of the topic, but with different query weight depending on the topic field they were extracted from. The weighting used was 4 for title terms, 2 for information need terms, and 1 for context terms. This strategy was named the "w" strategy.

In order to face the problem of the morphological variations of biological terms present in medical article abstracts, we expanded the queries obtained with the two strategies described above with the help of a script provided given to us by De Bruijn and Martin (2003) from the Institute of Information Technology at the National Research Council of Canada. This script creates morphological variations on the use of numerical symbols (Roman, Greek, contemporary), and by attaches or detaches numbers to or from the term they are related to. This expansion strategy was used for the TrecGen 2003 primary task and showed some success at improving the recall. The result is two non-expanded "ta" query strategies, "tn1" and "w1", and two expanded query strategies, "tn2" and "w2".

The construction of the MeSH queries was done manually. An expert from within the University read each one of the 55 topics (50 main topics plus the 5 sample topics) and chose the appropriate Main Headings with the assistance of the MeSH browser available on the Web. We also asked our expert to distinguish the most important Main Headings amongst those she had chosen for the topic. These Main Headings were very important in the sense that all documents deemed relevant to the topic should contain some or all of them. These we refer to as “Magic” Main Headings. This leads to a further retrieval strategy “ma” that only yields results that do in fact contain all the magic headings.

Similarly to the “ta” queries, we chose to have an expansion strategy for the MeSH queries. The MeSH vocabulary is organized into 15 trees. Each tree has a root that represents the most general concept of all the terms contained in the tree. Example of such concepts are “Anatomy” (tree A), “Chemicals and Drugs” (tree D) or “Geographic Locations” (tree Z). Further down the tree are represented concepts that are increasingly specific. Main Headings can appear several times in the hierarchy (1.8 times on average) and therefore can have several parents, siblings and children. However, multiple occurrences of the same Main Heading tend to be found in the same tree (85% of the time). Our expansion strategy involved looking up all the possible occurrences of a MeSH query term in the hierarchy and adding the parent(s) and the children of each to the query. We called the non-expanded MeSH query strategy and the expanded one respectively “m1” and “m2”.

4.2.3 Ranking strategies

Following the different types of queries we obtained three types of ranking: a “ta” ranking after querying the “ta” index with either strategy “tn” of “w” (expanded or not), a MeSH ranking after querying the MeSH index with strategy “m” (expanded or not), and a “magic” ranking after querying the MeSH index with the “magic” Main Headings only.

When the three types of ranking were used together, we used the “magic” ranking as a filter to the “m” ranking, i.e. documents retrieved in the “m” ranking but not retrieved in the “magic” ranking were removed from the “m” ranking. The resulting “m” ranking was then merged with the “ta” ranking as follows. Scores in each ranking were normalized with the best score in each ranking. A document found in only one ranking kept its original score in the merged ranking. A document found in both rankings would be integrated into the merged ranking with a score equal to the sum of its scores from both original rankings. The documents were then re-ranked according to their new score, and only the top 1000 were kept. To represent this new ranking, we used, for example, tn1_ma_m1, which describe the combination in which the non-expanded version of “tn” and “m” is used, a “magic” filtering of the “m1” ranking is completed, and the resulting “m1” ranking and “tn1” ranking are merged in the way we described above. The possible combinations including cases where the “ma” ranking was not used as a filter but as a proper MeSH ranking that was merged with a “ta” ranking (tn1, tn2, w1, w2) and cases where no filtering was used, and also where only one ranking was used on its own.

4.3 Genomic Results

The evaluation of these rankings or combinations of rankings uses the mean average precision over the 5 sample topics provided for the TrecGen 2004 ad hoc task. We used the trec_eval program along with the partial relevance judgements provided.

Table 2: Mean Average Precision and Recall for each Ranking Strategy

Ranking Strategy	Mean Average Precison	Recall (over the 5 queries)
m1	0.0201	64.5%
m2	0.0011	41.9%
tn1	0.0044	48.4%
tn2	0.0035	37.6%
w1	0.0035	49.5%
w2	0.0026	38.7%
m1_tn1	0.0201	59.1%
m1_tn2	0.0182	66.7%
m2_tn1	0.0065	47.3%
m2_tn2	0.0055	39.8%
m1_w1	0.0202	59.1%
m1_w2	0.0190	62.4%
m2_w1	0.0057	45.2%
m2_w2	0.0045	37.6%
m1_ma_tn1	0.0205	60.2%
m1_ma_tn2	0.0189	66.7%
m2_ma_tn1	0.0100	48.4%
m2_ma_tn2	0.0086	43%
m1_ma_w1	0.0206	59.1%
m1_ma_w2	0.0199	64.5%
m2_ma_w1	0.0092	49.5%
m2_ma_w2	0.0075	45.2%
Ma	0.0356	72%
ma_tn1	0.0215	48.4%
ma_tn2	0.0221	49.5%
ma_w1	0.0222	41.9%
ma_w2	0.0226	41.9%

Table 2 (above) shows the Mean Average Precision and Recall values for each ranking strategies described in section 4.2.3. We evaluated rankings from using each strategy on their own, the "ta" strategies (tn1, tn2, w1, w2) and the "m" strategies (m1, m2). Even if the performance was low for each one of these unique strategies, we notice a significant difference between the "m1" strategy and all the others. With 2.01% average precision and 64.5% recall, "m1" is the best of the unique strategies. The ranking obtained from the expanded query versions has lower performance than the non-expanded ones, especially "m2". Our morphological text expansion that we applied in "tn2" and "w2" decreases the Mean Average Precision (MAP) without improving the recall. Our policy of adding the parents and children of a MeSH Main Heading in "w2" also has a clear negative effect on both MAP and recall.

The rankings obtained from the combinations of the "text" or "ta" strategies with the MeSH or "m" strategies give better results on average than the unique strategies. Any combinations with "m1" would outperform any combinations with "m2" and given a combination with "m1" or "m2", the expanded "ta" strategies, "tn2" and "w2", yield lower MAP, as the early results seemed to suggest. However, in the case of "m1_tn2" and "m1_w2", we observe improvements in recall.

Next, consider text and MeSH strategy combinations where the "magic" filter was used. As explained above, this filtering process takes away from the MeSH ranking the documents which were not retrieved in the top 1000 documents ranked as a result of the "ma" query. This "ma" query contains descriptors that were judged as highly important by our expert. The results suggest that filtering always improves the MAP, and either improves the recall or leaves it unchanged. This suggests in turn that the "magic" descriptors provided by the expert were helping in the selection of relevant document and were not harming the recall level.

The rest of the table gives the results for the "ma" ranking on its own as well as the merging of the "ma" ranking with all the rankings resulting from "ta" or title/abstract queries. Although "ma" ranking was originally intended to be used as a filter to improve precision, it gave us the best result of all rankings and merging of rankings in both MAP and recall. As a consequence we proceeded to merge it with the "ta" strategies (tn1, tn2, w1, w2). The results show that those combinations yield a better MAP than any combination of strategies and unique strategies (except the "ma" ranking). However the recall is inferior to the one obtained when the "ta" strategies are merged with m1 (see m1_tn1, m1_tn2, m1_w1, m1_w2).

4.4 Genomic conclusions and future work

In order to select the two runs for submission, we decided to take a broad look at the results. Table 1 allows us to compare the different elements of the ranking combinations more than it allows us to choose a particular combination, given the partial nature of the relevance judgments. We thus conclude that the expanded text queries ("tn2" and "w2") did not seem to work and that the terms extracted from the title and information need fields ("tn") of the topics were better query terms than the ones extracted from all the topic fields with weights ("w"). We inferred that the "tn1" ranking strategy should be part of the run submitted. The expanded version of the MeSH queries damaged the performance and the magic queries gave the best performance in both Mean Average Precision and recall. We were confident that a combination of "tn1" and "ma" would give satisfactory results and we submitted the "ma_tn1" combination as our first choice. As a second choice we submitted the "ma" ranking on its own because it's high performance with the 5 sample topics although we had little confidence that it would perform well over the 50 topics.

The "bag of descriptors" representation of documents does not allow us to calculate a suitable similarity measure between documents that have few terms in common but which terms are related in the MeSH trees. We plan to make full use the MeSH hierarchy and experiment with several tree similarity measures in order to build a better document representation and more appropriate similarity measures. We also want to exploit the distinction made in the MEDLINE manual indexing process between "central concepts" and "peripheral concepts" and integrate the use of "qualifiers" during the document representation building process (as discussed in section 4.2.1).

5. NOVELTY TRACK ACTIVITIES

The TREC 2004 Novelty Track was structured into three distinct tasks. Participants in the first task were required to retrieve a subset of the twenty-five relevant documents for each of the fifty topics. Once this task was performed, participants were required to extract from within those retrieved documents all the sentences that were relevant to each of the fifty topics. Finally, given this list of retrieved relevant sentences, which were ranked in a predetermined order, participants were required to select a subset of sentences that provided *novel* or *new* information on the topic.

A sentence S_n is considered novel if and only if its similarity with all previously seen and highlighted novel sentences S_1…..S_{n-1}, is below a certain threshold.

As this is the first year that DCU has participated in the Novelty Track, we first investigated various methods employed in the previous two years the Novelty Track has run. We present here the algorithms we developed to complete Task 1 and Task 2. We built three models; the first focused on retrieving the twenty-five documents that were relevant to each topic; the second focused on retrieving relevant sentences from this list of retrieved documents to satisfy each individual topic; the third focused on the detection of novel sentences from this relevant list.

In Task 1 we were given a collection of documents and are asked to retrieve all new and relevant information for each of the fifty topics. We used *Físréal*, an information retrieval system developed by the CDVP for the Terabyte Track (described in section 2), as a basis for our experiments, which was configured to use BM25 weighting. In Task 2 we were given the relevant sentences and are asked to find the novel sentences. This allows us to accurately assess our novelty models, as we are not depending on a system's ability to retrieve sentences. In this task we regarded sentences as documents, and developed two formulas, which exploit traditional document similarity methods.

5.1 Task1 : Redundancy

Traditionally in Information Retrieval, implementation and experimentation of retrieval algorithms have dealt with large collections, containing many documents. However, the Novelty Track explores various methods to find relevant *sentences* within a document. It has been shown over the last two years that sentence retrieval is more difficult to accomplish than traditional document retrieval as there are less words in each document, and hence less accuracy and more ambiguity (Min et al., 2003). Our approach to improving performance was to extend the document and the query data using Dekang's (http://www.cs.ualberta.ca/~linkdek/downloads.htm) dictionary of both similarity and proximity term dependency. This dictionary was successfully employed to improve system performance by THUIR at TREC2003. We submitted five runs which we discuss later.

5.1.1 Stage One

This involved the detection of the relevant twenty-five documents in the collection supplied by NIST for each of the fifty individual topics. We automatically generated a query for each topic, which concatenated together the title, the description and the narrative (negative information from the narrative was automatically removed) from the TREC query. The top twenty-five highest ranked documents returned from Físréal were chosen as our relevant documents.

5.1.2 Stage Two

This involved the detection of relevant sentences for each of the fifty topics, where our input was the set of documents retrieved in stage one. We experimented with different combinations of query and document expansion, utilising Dekang's dependency dictionary. In Run 1 (cdvp4CnS101), we queried the system using a long query against document expanded using the similarity dictionary. In Run 2 (cdvp4QePnD2), we examined the effects of expanding the query alone using the proximity dictionary. Run 3 (cdvp4CnQry2), used traditional searching with no performance enhancement on either the query or document. In Run 4 (cdvp4QePDPC2), we experimented with a combination of query and document expansion using the proximity dictionary and in the final Run 5 (cdvp4QeSnD1), we investigated the benefits of using just query expansion which utilised term similarity dictionary.

5.1.3 Stage Three

This stage takes as input the ranked list of relevant sentences produced in the previous step, and returns only those sentences that provide new or novel information for each topic. This novelty model cannot be accurately assessed as its overall performance is dependent on the results of both stages one and two, in which the initial detection of relevant data is carried out. With so many contributing factors, it is difficult to decipher whether the novelty algorithms are performing well. The first method used in Task 1 to detect sentences that were novel was based on the *UniqueHistory* formula (Section 5.2) using a threshold of 3. The second method used was the *NewSentenceValue* formula (Section 5.2) using a threshold of 0.08.

5.2 Task 2 : Novelty

In this task, participants were given an ordered list of relevant sentences for each of the fifty topics and were required to detect a subset of this ranked list, which will provide novel or new information to the users. We evaluated the performance of our system using variations of three different formulas *ImportanceValue* (Section 5.2.1), *NewSentenceValue* (Section 5.2.2) or *UniqueHistory* (5.2.3). We submitted a total of five runs (see Table 3). Given a sentence S_n in the ordered list of relevant sentences, we evaluated the number of unique words U_s that occur in the set of words S_s in the current sentence S_n, against an accumulating list of all unique words U_H encountered to this point (for a particular topic). We then evaluated U_s using one of the three formulas, and assigned to each sentence its novelty score. If this score was above a predefined threshold, the set of unique terms U_s was added to the history set U_H and the sentence was added to the list of novel sentences to be returned to the user. The Baseline run (*cdvp4NSnoH4*) used the *UniqueHistory* formula however we did not keep an accumulated history set of all the previous sentences. The results were very poor for this run, which in retrospect, was not surprising.

Definitions: Let

Current sentence = $\mathbf{S_n}$ *Set of words for the current sentence=* $\mathbf{S_s}$	$\|\mathbf{S_s}\|$ *the size of the current sentence set=* $\mathbf{N}$
All previously seen sentences= $\mathbf{S_1 \ldots\ldots S_{n-1}}$	
Set of Unique Words in current sentence= $\mathbf{U_s}$	$\|\mathbf{U_s}\|$ *the size the of unique set =* $\mathbf{n}$
Accumulating set(History Set) of unique words over a particular topic = $\mathbf{U_H}$	
Term Frequency = tf	*Inverse document frequency =* idf
Unique terms in a sentence = $tu \in U_s$	
Terms of the History set = $th \in U_H$	*Current terms of a sentence=* $tc \in S_s$

5.2.1 The Importance Value

This function (see Formula 2) exploits the properties of a term from both within a sentence and the collection of sentences from which it originates. It models the assumption that a term with a high term frequency (*tf*) and a high inverse document frequency (*idf)* would most likely be a valuable term in providing new and valuable information about a topic. A sentence assigned a novel score, higher than a predefined threshold, is considered a novel sentence.

$$\left(\sum_{i=1}^{n} tf_{tu_i} \cdot \sum_{i=1}^{n} idf_{tu_i}\right) \cdot \frac{1}{N} \qquad (2)$$

5.2.2 The NewSentenceValue

The NewSentenceValue formula (see Formula 3) exploits the characteristics of a term within a collection of sentences. We analyse the importance-value of the unique terms U_s and assess their overall benefit, in contributing novel information to the user. A sentence with a novel score above a predefined threshold is considered a novel sentence.

$$\frac{\sum_{i=1}^{n} idf_{tu_i}}{\sum_{j=1}^{m} idf_{tc_j}} \cdot \sum_{k=1}^{l} idf_{th_k} \qquad (3)$$

5.2.3 The UniqueHistory

The novel score of a sentence was determined by the evaluating the number of unique words U_s that occurred in the sentence word set against an accumulating list of all unique words U_H encountered to this point (for a particular topic). If the number of unique words exceeds a particular threshold then the sentence was considered novel. This is a crude way to determine novelty but as the results show it is a method which gives comparable results.

5.3 Novelty Results

It can be observed (Table 3) that all relevance runs performed equally well and all were above the mean results for the track. The highest F-scores (marginally) were achieved by applying query expansion, using the term proximity only (run cdvp4QePnD2). When we compare the *novel* results of Task 1 to the overall results of Task 2, it can be seen that the performance of Task 1, in which used a combination of two methods, is substantially lower than Task 2. These results could be caused by the dependency of Task 1's novelty system on relevant results.

Table 3: Task1 - Results , Relevance and Novelty

Task1 Relevance	Run	Precision	Recall	F-Score
Average(mean)				0.388
CnQueryDocExpSim	cdvp4CnS101	0.30	0.73	0.402
QeProxnoDocExpUniqclr	cdvp4QePnD2	0.31	0.71	0.406
CnQueryUniqHistory	cdvp4CnQry2	0.32	0.67	0.405
QeProxDocExpProxUnqH	cdvp4QePDPC2	0.29	0.78	0.397
QeSnoDocExpNewSenVal	cdvp4QeSnD1	0.31	0.72	0.404

Task1 Novelty	Precision	Recall	F-Score
			0.203
	0.13	0.71	0.201
	0.13	0.70	0.205
	0.15	0.55	0.221
	0.14	0.63	0.212
	0.14	0.64	0.216

Table 4 displays the results of task2. It can be clearly seen that the performance of run (*cdvp4NTerFr1*) implementing the *ImportanceValue* formula at threshold 1.5 was better than that of other runs and all runs, except the Baseline, were above the mean.

Table 4: Task2 Results , Precision, Recall and F-score

Task2 Novelty	Run	Precision	Recall	F-Score
Average(mean)				0.597
ImportanceValue >1.5	cdvp4NTerFr1	0.49	0.90	0.622
ImportanceValue >3.5	cdvp4NTerFr3	0.51	0.83	0.616
NewSentenceValue>0.08	cdvp4NSen4	0.47	0.91	0.609
UniqueHistory >3	cdvp4UnHis3	0.50	0.84	0.615
Baseline	cdvp4NSnoH4	0.38	0.49	0.383

Analysing the performance of runs by topic type, we find that in detecting relevance (task1, Table 5) our F-scores show that there is a variance between the systems performance on event and opinion topics. This variance can also been seen in detecting the novel sentences between opinion and event topics of task1.

Table 5, Task1 Performance of the system over Event and Opinion Topics

Task1 Relevance	Run	Events	Opinion
Average(mean)		0.472	0.304
CnQueryDocExpSim	cdvp4CnS101	0.473	0.330
QeProxnoDocExpUniqclr	cdvp4QePnD2	0.477	0.335
CnQueryUniqHistory	cdvp4CnQry2	0.472	0.339
QeProxDocExpProxUnqH	cdvp4QePDPC2	0.471	0.322
QeSnoDocExpNewSenVal	cdvp4QeSnD1	0.475	0.333

Task1 Novelty	Event	Opinion
	0.251	0.148
	0.248	0.153
	0.252	0.158
	0.267	0.176
	0.263	0.161
	0.266	0.167

In Task2 (see *Table 6*) where the relevant judgments have been provided we notice that the performance of our system in detecting opinions and events is very comparable and once again, all our results except baseline are above the mean..

Table 6, Task2 Performance of the system over Event and Opinion Topics

Task2 Novelty	Run	Events	Opinion
Average(mean)		0.598	0.596
ImportanceValue >1.5	cdvp4NTerFr1	0.634	0.609
ImportanceValue >3.5	cdvp4NTerFr3	0.624	0.607
NewSentenceValue>0.08	cdvp4NSen4	0.612	0.606
UniqueHistory >3	cdvp4UnHis3	0.626	0.603
Baseline	cdvp4NSnoH4	0.375	0.390

6. ACKNOWLEDGEMENTS

The support of the Informatics Directorate of Enterprise Ireland is gratefully acknowledged. Part of this material is based on work supported by Science Foundation Ireland under Grant Nos. 03/IN.3/I361. The work described in section 4 is part of project GenIRL that is funded by Enterprise Ireland under the Basic Research Grants Scheme, project number SC-2003-0047-Y. We would like to thank our biology domain expert Fionnuala McDyer for helping us by manually annotating each topic with terms from the MeSH controlled thesaurus.

7. REFERENCES

Blott S, Gurrin C, Jones G J F, Smeaton A F and Sodring T (2003). “On the use of MeSH headings to improve retrieval effectiveness”. In Proceedings of the 12th TREC Conference, Gaithersburg, Md, November 2003.

De Bruijn B and Martin J (2003). Finding Gene Function using LitMiner. *The Twelfth Text REtrieval Conference, TREC 2003,* Gaithersburg, MD. NIST.

Craswell N and Hawking D (2003). Overview of the TREC 2003 Web Track, Proceedings of the Twelfth Text REtrieval Conference, pg78-92, 2003.

Freyne J, Smyth B, Coyle M, Balfe E, and Briggs P (2004). Further Experiments on Collaborative Ranking in Community-Based Web Search. In: Artificial Intelligence Review: An International Science and Engineering Journal, 21 (3-4):229-252, June 2004.

Funk M E and Reid C A (1983). Indexing consistency in MEDLINE. *Bulletin of the Medical Library Association*. 71(2):176–83.

Min Z, Lin C, Liu Y, Zhao L, Ma L, Ma S (2003). THUIR at TREC 2003: Novelty, Robust, Web and Hard, State Key Lab of Intelligent tech & Sys, CST Dept, Tsinghua University, Beijing 100084, China

Robertson S E, Walker S, Hancock-Beaulieu MM and Gatford M (2004). Okapi at TREC-3. Proceedings of the Third Text REtrieval Conference, pg109–126, 1994

Srinivasan P (1996). Query Expansion and MEDLINE. *Information Processing and Management,* 32(4): 431-443.