The QMUL Team with Probabilistic SQL at Enterprise Track

Thomas Roelleke Elham Ashoori Hengzhi Wu
Zhen Cai

Queen Mary University of London

Abstract

The enterprise track caught our attention, since the task is similar to a project we carried our for the BBC. Our motivation for participation has been twofold: On one hand, there is the usual challenge to design and test the quality of retrieval strategies. On the other hand, and for us very important, the TREC participation has been an opportunity to investigate the resource effort it requires to deliver a TREC result.

Our main findings from this TREC participation are: 1. Through the consequent usage of our probabilistic variant of SQL, we could describe retrieval strategies within a few lines of code. 2. The processing time proved sufficient to deal with the collection. 3. The abstraction-oriented data modelling layers of our HySpirit framework enable relatively junior researches to explore a TREC collection and submit runs. 4. For the less complex retrieval tasks (discussion search, known-item search), minimal resources lead to acceptable results, whereas for the more complex retrieval tasks (expert search), inclusion and combination of all available evidence appear to significantly improve retrieval quality.

1 Introduction

The enterprise track at TREC caught our attention, since at QMUL we carried out an expert search project with the BBC. In that project, several highly heterogeneous data sources needed to be explored for creating expert profiles, and the customisable ranking of experts in a number of different contexts was required.

Through the BBC and several other projects, we learned that the ability to adapt to the requirements of the customer is crucial. Also, alive data sources are highly complex and comprise both SQL-like and text-like data. Because of the many heterogeneous systems involved, the semantic data exploration is a highly challenging task.

Because of these requirements in real-world enterprise search, our approach is to use an integrated DB+IR technology for building customised search systems. Our system (referred to as HySpirit) is built on the results reported in [FR97] and [FGR98]. Since 2000, improvements in usability, robustness, efficiency and scalability have been achieved, most of which have been not reported yet in academic publications.

This TREC report contains a description of our probabilistic SQL (PSQL) layer. This SQL-like approach to information retrieval is for many comparable to what is presented in [GF98], who report in the respective section that probably [Bla88] and other authors in the late 80s (following the early work [Cra81]) applied SQL for text retrieval. As we will point out in this paper, our SQL-like approach to information retrieval provides a high abstraction layer, i.e. different from [GF98] and others, in which the scoring function is an explicit piece of math, our approach is to provide concepts of probability theory. The list of approaches to apply DB technology for text search or to integrate DB technology with text search is endless, and just to mention few: [DM97] is an attempt in the DB field to integrate DB-like and text-like search. [Hie03] is an IR approach for using DB-like technology for XML retrieval. [RR02] is an excellent contribution in pointing out that the usual aggregation functions of SQL in a probabilistic framework yield the expectation value of attribute values. This underlines that the bare SQL aggregation functions are different from the probability aggregation in a probabilistic database. Other recent developments in the field include [DS04] and [LCIS05]: The DB-approach to ranking with SQL emphasizes efficient query processing. Also, the DB attention to the field shows that PSQL is not only from an expressiveness point of view, but also from the support and attention it gets in the DB research, a promising candidate for solving future retrieval tasks.

Whereas the application of SQL for text retrieval is somewhat intuitive (just model a collection as a relation "index(term, document)" and perform SQL statements), scalability is a significant drawback of such an approach. Of course, DB vendors are aware of this, and therefore offer SQL/Text, SQL/Multimedia, and other SQL variations, in which special data types and operators take care of text and multimedia objects, i.e. an inflation of tuples is avoided, and the database systems provide special operators and indexing structures to deal with large attribute values. This approach is sufficient if a customer is satisfied with the burned-in retrieval strategies defined for the special attribute values.

However, customisation of retrieval strategies is the requirement in enterprise search environments, and we wish to combine the evidence from structured and unstructured data sources in a well-defined framework. [Haw04] is an inspiring contribution on what makes enterprise search different from "normal" search.

For the customisation typical for enterprise search, probabilistic relational algebra/SQL provides the required flexibility and expressiveness.

Search tasks such as discussion search, known-item search, and expert search involve the implementation of different (customised) retrieval strategies. Rather than using an existing IR system and playing with the parameter setting or modifying the source code, our approach is to use strictly the underlying concepts of database technology. As there are expressiveness and scalability problems when applying classical SQL in large-scale IR applications, we developed a new probabilistic SQL platform with a particular emphasis on achieving an integrated DB+IR platform.

For making explicit what distinguishes a DB+IR SQL from an SQL-based IR implementation, we summarise in section 2 the intuitive way of an SQL-based IR implementation, and show in section 3 the probabilistic extension. Then, in section 4, we present our results for the discussion and known-item search tasks; for these two tasks, we applied the same strategies. For expert search, we defined tailored strategies in PSQL, and the strategies and results are discussed in section 5.

This is the first year, QMUL participated in TREC. We mainly focused on the power of our retrieval engine in building an ad-hoc search system with minimal resources. Since one of our master students became interested in the enterprise track, we gained the required resources and could participate. For the expert search task, we restricted the search/evidence space to the emailing lists only, i.e. we did not consider the evidence available in web pages, and the other sub-collections of enterprise track.

2 Modelling Information Retrieval with SQL

2.1 Representation of collection and query

The basic idea of modelling text document retrieval with SQL is to represent the collection and the query in relations (tables) such as the following:

collection(term, doc)
query(term)

For example:

Relation *collection*	
term	*document*
sailing	doc1
boats	doc1
sailing	doc2
sailing	doc2
boats	doc2
east	doc2
coast	doc2
sailing	doc3
sailing	doc4
east	doc5

The example is constructed so that the reader can easily follow the computations in the following sections. The collection is represented in ten tuples. We consider five documents, and four terms. The tuple "(sailing,doc2)" occurs twice, all other tuples occur just once.

Of course, the relational representation of this content-oriented index is expensive with respect to disc space, and the large amount of tuples in a relation is a problem for classical relational databases. For this, the HySpirit system provides special operations and index structures, and some techniques are indicated in section 3 on probabilistic SQL.

From the basic relation representing the content of the documents (collection), document and term frequencies are derived in the following sections.

2.2 Document and term frequencies

For illustrating the description of the inverse document frequency, we create first a table doc_freq(term,value) in which doc_freq.value represents the number of documents in which the term occurs.

Consider the following SQL statement and its result in relation doc_freq:

```
INSERT INTO doc_freq
SELECT term, count(DISTINCT doc)
FROM collection
GROUP BY term;
```

Relation *doc_freq*	
term	*value*
sailing	4
boats	2
east	2
coast	1

Next, we derive the number of documents, and compute the idf-value for each term.

```
INSERT INTO number_of_docs
SELECT count(DISTINCT doc)
FROM collection;
```

```
INSERT INTO idf
SELECT term,
log(number_of_docs.value / doc_freq.value)
FROM doc_freq, number_of_docs;
```

So far for the computation of idf-values. Next, we compute the term frequency (tf) values. The within-document term frequency is expressed via the following statement:

```
INSERT INTO term_freq
SELECT term, doc, count(*)
FROM collection
GROUP BY term, doc;
```

For our toy collection, we obtain:

Relation *tf*		
term	*doc*	*value*
sailing	doc1	1
boats	doc1	1
sailing	doc2	2
boats	doc2	1
east	doc2	1
coast	doc2	1
sailing	doc3	1
sailing	doc4	1
east	doc5	1

More complex but more effective methods known from BM25, in which the term frequency is reflected by the factor $n(t)/(K+n(t))$ can be expressed as well in SQL; just the aggregation expression becomes more complex to formulate.

2.3 Retrieval

For retrieval, basically, the relation representing the query is joined with the relations representing the document and term frequencies.

```
SELECT tf.doc, sum(idf.value * tf.value)
FROM query, idf, tf
WHERE query.term = idf.term
AND query.term = tf.term
GROUP BY tf.doc
```

The above SQL-based modelling of information retrieval appears intuitive to the database expert, though, despite the welcome abstraction provided by SQL, SQL is not used for implementing large-scale information retrieval tasks and systems. Why? Because the abstraction is actually not great. The SQL programmer has to specify the computation of frequencies and the retrieval status value in a rather physical way. Secondly, the SQL-based modelling of IR has significant scalability problems because of the group-by and aggregation operations involved. These operations do not scale for the basic modelling shown above, since with each new tuple arriving in the collection table, the frequencies need to be computed. Even if this could be solved (for example, by using special indexes or materialised views as they are applied in data-warehousing when aggregating large amounts of data), the computation of the retrieval status value involves a non-scalable distinct selection. The more document a term retrieves, the longer the distinct selection will take to be computed.

For improving abstraction and scalability of SQL, we developed a probabilistic SQL variant based on [FR97].

3 Modelling Information Retrieval with Probabilistic SQL

3.1 Representation of collection and query

Collection and query are represented in probabilistic SQL as they are represented in standard SQL, see section 2.1.

3.2 Document and term frequencies

In probabilistic SQL, due to the probabilistic paradigm, we estimate probabilities from frequencies, and for this estimation, probabilistic assumptions are specified. As a first example, consider the creation of a disjoint space of documents.

```
CREATE VIEW doc_space AS
SELECT DISTINCT doc
FROM collection
ASSUMPTION DISJOINT;
```

Through the above view creation, the probabilistic table doc_space contains the following tuples:

doc_space	
0.2	doc1
0.2	doc2
0.2	doc3
0.2	doc4
0.2	doc5

The probabilities are not usual attribute values when working in the paradigm of probabilistic SQL (PSQL), i.e. PSQL manipulates the tuples based on the algebra operations but the PSQL programmer has no direct influence on the arithmetics of probabilities.

Next, we create views for reflecting the document frequency (df) and inverse document frequency (idf) of terms.

```
CREATE VIEW df AS
SELECT DISJOINT term
FROM distinct_collection, doc_space
WHERE
distinct_collection.doc = doc_space.doc;
```

The table distinct_collection contains the distinct tuples of table "collection". The join of distinct_collection and doc_space yields the following relation for the document frequency:

df	
0.8	sailing
0.4	boats
0.4	east
0.2	coast

Given the document frequency, the following view creation allows for assessing a term probability which corresponds to the idf of a term:

```
CREATE VIEW idf AS
SELECT *
FROM df
ASSUMPTION MAX_IDF;
```

The assumption max_idf normalises the idf values by dividing each idf-value with the maximal idf-value. Here, $idf(coast) = -\log 0.2$ is maximal, and we obtain:

idf	
0.139	sailing
0.569	boats
0.569	east
1.0	coast

After having described the inverse document frequency, we describe next the within-document term frequency. The intuitive n/N-based estimate for the within-document term frequency is described as follows:

```
CREATE VIEW tf AS
SELECT DISJOINT term, document
FROM collection
ASSUMPTION DISJOINT
EVIDENCE KEY (2);
```

Note that in contrast to the creation of the df-table, we base the tf-view on the *non-distinct* relation "collection", so to capture the non-distinctness of tuples in the probabilities generated for the tf-view. The evidence key is a specialty of our probabilistic SQL: Here the probabilities shall be normalised for each document, hence the evidence key contains the second attribute of the target list. The above SQL statement yields the following table:

tf		
0.5	sailing	doc1
0.5	boats	doc1
0.4	sailing	doc2
0.2	boats	doc2
0.2	east	doc2
0.2	coast	doc2
1.0	sailing	doc3
1.0	sailing	doc4
1.0	east	doc5

Results from earlier projects (INEX, BBC) and research findings (cite BM25) allow to draw the conclusion that the disjointness-based $P(t|d)$ is less effective than the Poisson-based estimate.

A Poisson-based estimate is described as follows in probabilistic SQL:

```
CREATE VIEW tf AS
SELECT DISJOINT term, document
FROM collection
ASSUMPTION POISSON
EVIDENCE KEY (2);
```

The semantics of the PSQL statements involving ASSUMPTIONS and EVIDENCE KEYS are described in the patent application [Roe03]. Recent attempts to publish the semantics of PSQL failed so far, but the semantics will be formally specified in future publications. The above statements are actually abbreviations of the probabilistically correctly extended statements.

3.3 Retrieval

For retrieval, we create a view "weighted_qterm" of weighted query terms:

```
CREATE VIEW weighted_qterm AS
SELECT term
FROM query, idf
WHERE query.term = idf.term;
```

Given the idf-weighted query terms, the join of weighted query terms and tf-based document index yields a tf-idf baseline retrieval result.

```
SELECT DISTINCT doc
FROM weighted_qterm, tf
WHERE weighted_qterm.term = tf.term;
```

The distinct aggregation of document tuples is here based on an independence assumption. Alternatively, we can normalise the weighted query terms, and form as usual a weighted sum of tf-idf values; this corresponds in probabilistic SQL to a disjointness assumption.

For enterprise track, we experimented with a number of different probabilistic aggregations. This is reflected in the next sections (sections 4 and 5) in which we report on discussion, known-item and expert search.

4 Discussion and Known-Item Search

For known-item and discussion search, we ran several strategies where we varied the probabilistic aggregation of term probabilities, and applied for each probabilistic variation a stemmed and an unstemmed match.

1. idf-based with independence assumption on query terms (section 4.1)
2. tf-idf-based with independence assumption on query terms (section 4.2)
3. tf-idf-based with disjointness assumption on query terms (section 4.3)

The different strategies are reflected in our run identification consisting of "qmir", and 'dj' or 'dt' for disjoint and distinct, and 's' or 'u' for stemmed or unstemmed.

Since for both, discussion and known-item search, the requested items were emails, we did not customise the strategies for the respective tasks. Actually, we expected for discussion search that the retrieved items should have been email threads rather than individual emails, in which case different retrieval strategies for discussion search and known-item search would have been reasonable.

The following sections describe briefly the rationales of the strategies, where our main aim is to investigate the effect of idf and probabilistic assumptions for the aggregation of the evidence each query term contributes to the retrieval status value (RSV).

4.1 Strategy 1: idf-based, independent query terms

As the very first strategy, a simple idf-based ranking method was applied. This strategy involves the definition of a view for the idf relation:

```
CREATE VIEW df AS
SELECT term FROM collection;

CREATE VIEW idf AS
SELECT term FROM df
ASSUMPTION MAX_IDF;
```

Apart from the ASSUMPTION clause, this is SQL as usual. The view for the document frequency (referred to as df-view) is here a helper for defining the idf-view. The df-view is inserted into the view definition of the idf, and based on a special indexing structure for probabilistic relations, we retrieve idf-based term probabilities in O(1).

Given the idf-view, we define the weighted query terms as the join of query terms and idf:

```
CREATE VIEW weighted_query AS
SELECT term FROM query, idf
WHERE query.term = idf.term;
```

Now, in relation weighted_qterm, each query term has a probability reflecting the idf, i.e. probabilities are high for rare terms, and small for frequent terms.

The weighted query terms are then matched against the term index, and the probabilities need to be aggregated for obtaining a retrieval status value.

```
SELECT DISTINCT doc
FROM weighted_query, doc_index
WHERE weighted_query.term = collection.term;
```

For a distinct selection, per default, an independence assumption is applied.

The relation “collection” is non-distinct, i.e. the join delivers a tuple for each occurrence of a query term in the collection. Through this, the independent projection aggregates the idf-based term probability proportional to the number of times a query term occurs in a document. From this point of view, this first simple strategy is not just an idf-only strategy, since the within-document term frequency has an effect on the RSV.

Let $P(t|q, c)$ be the idf-based probability of a term given, given query q and collection c. $P(t|q, c)$ is the probability function in relation weighted_qterm, and the function is defined through the above SQL as follows:

$$P(t|q, c) := \begin{cases} P_{idf}(t|c) & \text{if } t \in q \\ 0 & \text{otherwise} \end{cases}$$

Let d be a document. The RSV implemented by the above SQL based on an independent aggregation of probabilities is as follows:

$$RSV(d, q) = 1 - \prod_{t \in d} (1 - P(t|q, c))$$

Frequent terms with $P(t|q, c) \approx 0$ have little effect on the RSV, whereas rare terms with $P(t|q, c) >> 0$ have a strong effect. With each occurrence of t in d, the idf-based probability $P(t|q, c)$ is taken into account. One could interpret the above RSV as a probabilistic disjunction of query term occurrences in the respective document, where each occurrence has the idf-based probability $P(t|q, c)$.

If there exists one $P(t|q, c) = 1$, then $RSV(d, q) = 1$. The explanation of this is as follows: If the query involves a term which occurs in just one document, then $P(t|q) = 1$. The document retrieved with respect to the query is assumed to be top relevant since the term occurs only in this particular document. For example, let the query be "t1 t2 t3". Then, if the query terms are maximal informative, each document containing one of the terms will be judged top relevant. The obvious problem with a non-weighted probabilistic disjunction is of course that there is no discrimination between documents that contain one, two or three of the query terms. To counter for this, we apply a normalisation on query terms:

```
CREATE VIEW weighted_query AS
SELECT term FROM query, idf
WHERE query.term = idf.term
ASSUMPTION DISJOINT;
```

Here, the assumption disjoint leads to a probability distribution in relation weighted_query where the tuple probabilities sum up to 1.0.

For the selection (retrieval) of documents, a disjointness assumption is not applicable, since the relation “collection” is not distinct. This is exactly what the next strategies address: Instead of the simple retrieval based on the raw collection representation, we create for the next strategies a distinct representation of the collection in a tf-based relation with terms and documents, where the probabilities reflect the multiple (non-distinct) occurrences of terms.

4.2 Strategy 2: tf-idf-based, independent query terms

Different from strategy 1, we apply now a distinct representation of the collection. We create a tf-based representation with the following view:

```
CREATE VIEW tf AS
SELECT DISJOINT term, document
FROM collection
ASSUMPTION POISSON
EVIDENCE KEY (2);
```

When retrieving documents, we match now the idf-based query terms with the tf-based representation of the collection:

```
SELECT DISTINCT document
FROM weighted_query, tf
WHERE weighted_query.term = tf.term;
```

This SELECT statement implements a tf-idf-based retrieval function. Here, the aggregation of tf-idf-based probabilities is based on an independence (the default) assumption.

4.3 Strategy 3: tf-idf-based, disjoint query terms

This strategy is virtually the same as the previous one, except that it assumes disjointness of the weighted query term. As in section 4.2, first, the query term are weighted according to their probability values estimated by IDF. Then we assume that the query terms are disjoint, effectively normalizing the probabilities, so that all of them added up to one. Finally, these probabilities were combined with TF probabilities to form the ranking probability for the retrieved documents.

```
SELECT DISJOINT document
FROM weighted_query, tf
WHERE weighted_query.term = tf.term;
```

4.4 Submissions and Results

Table 1 shows an overview over our submitted runs. The run identifier is constructed as follows: qmir-[task]-[distinct|disjoint]-[(s)temmed|(u)nstemmd]. There is a task identifier 'ki' for known-item search, and 'ex' for expert search, no identifier for discussion search, as these were the first runs submitted. The abbreviation 'dt' stands for distinct, and 'dj' stands for disjoint.

Table 1 shows the expert search runs as well. We will discuss expert search modelling and results in section 5.

Table 2 shows the results for discussion search and known-item search.

The disjointness-based tf-idf strategy, unstemmed, shows the best retrieval quality. Here, the tf-based probabilities were estimated based on the Poisson-like curve $tf/(avg(tf) + tf)$. No normalisation and no special parameter setting has been applied; the whole retrieval function is clean of parameters.

The result for the discussion search, namely that the disjointness-based tf-idf strategy performs best, is the expected result. The somewhat surprising result is that the independence-based strategies are pretty close in retrieval quality. Remember that for the independence-based strategies, no tf-based representation of the collection was created. This is nice since it facilitates update operations in dynamic environments, where this result allows us to maintain only the raw representation of the collection and save on the tf-based representation.

For the known-item search, the conclusion is slightly different, though the marginal difference in MRR is hardly significant. Here, the disjointness-based strategy is slightly worse than the independence-based strategy. In average, if found, the known item is within the first 3-4 hits. The mismatch appears to be rather high. Both submitted runs are unstemmed, that might explain the high percentage of not found. For one strategy, there are 28 queries with not found, and for the other, there are 29 queries. This asks for an explanation, since both strategies use exactly the same termspace, only the probability aggregation differs. The submissions for known-item search with stemming unfortunately got lost somewhere between QMUL and TREC systems.

5 Expert Search

For this task, we implemented basically two strategies: A baseline strategy in which we aggregate the email RSV's per author, and a more refined strategy in which we consider the global and local author contribution, i.e. how many emails does someone write overall (global contribution), and how many emails does someone write for the query topic (local contribution).

5.1 Strategy 1: aggregation of email RSV's

In this strategy, experts are considered as those people who authored an email. In previous projects, we actually discarded this information, as it appears more appropriate to identify experts based on the

Submissions		
Run identifier	Task	method
qmir-dj-u	Discussion Search	tf-idf, disjoint query terms, unstemmed
qmir-dj-s	Discussion Search	tf-idf, disjoint query terms, stemmed
qmir-dt-u	Discussion Search	idf, independent query terms, unstemmed
qmir-dt-s	Discussion Search	idf, independent query terms, stemmed
qmir-ki-dj-u	Known-Item Search	tf-idf, disjoint query terms, unstemmed
qmir-ki-dj-s	Known-Item Search	tf-idf, disjoint query terms, stemmed
qmir-ki-dt-u	Known-Item Search	idf, independent query terms, unstemmed
qmir-ki-dt-s	Known-Item Search	idf, independent query terms, stemmed
qmir-ex1	Expert Search	GCF low, LCF low
qmir-ex2	Expert Search	Aggregation of independent email evidence
qmir-ex3	Expert Search	GCF low, LCF max
qmir-ex4	Expert Search	GCF low, LCF high

Table 1: Submitted runs

Discussion Search			
Run identifier	MAP	Bpref	P@10
qmir-dj-u	**.2860**	**.3017**	**.4695**
qmir-dt-s	.2723	.2777	.4136
qmir-dt-u	.2706	.2804	.4576

Known-Item Search		
Run identifier	MRR (if found)	# Not Found
qmir-ki-dj-u	0.360	28 (22.4%)
qmir-ki-dt-u	0.367	29 (23.2%)

Table 2: Results for Discussion Search and Known-Item

emails they received. Since the recipient information is not exploitable in the TREC test data (emails go to lists), we decided as a baseline strategy to retrieve the emails for a topic, and then aggregate retrieval status values per author to obtain a ranking of experts. This is very much like passage retrieval ([Cal94]). The processing steps are as follows:

1. Retrieve top-N emails in the same way as discussion or known-item search strategies
2. Join emails with author relation to obtain authors
3. Rank authors based on the aggregation of email retrieval status values (RSV's)

The strategy was submitted under id **qmirex2**.

5.2 Strategy 2: local and global author contribution

The processing steps for this strategy are as follows:

1. Retrieve top-N emails
2. Join emails with author relation to obtain authors
3. For each author, compute the collection-wide (global) email frequency, and compute the local email frequency, i.e. the frequency in the retrieved emails. We refer to the collection-wide frequency as Global Contribution Frequency (GCF) (in the collection c), and we refer to the frequency in the retrieved emails as Local Contribution Frequency (LCF) (in the set r of retrieved documents). The frequencies are as follows:

$$\begin{aligned} GCF &= \frac{occ(author, collection)}{N_D(collection)} \\ LCF &= \frac{occ(author, retrieved)}{N_D(retrieved)} \end{aligned}$$

   Here, c and r are set of documents, and $N_D(c)$ and $N_D(r)$ are the numbers of documents in the respective set.

4. Rank authors based on the global and local contribution.

We submitted runs in which we tested various combinations of GCF and LCF.

5.2.1 GCF low, LCF high

qmirex4: We apply a ranking similar to what tf-idf does for terms: If an author occurs frequently among the retrieved emails (LCF high) and if an author occurs rarely in the collection (GCF low), then this author is likely to be an expert in the topic. Thus, we obtain an author frequency per topic (analogous to the classical tf, where we model the term frequency per document), and we obtain an idf of each author (analogous to the classical idf, where we consider the idf of a term).

For implementing this strategy, we join the email retrieval result with the author names.

```
– Get the authors of retrieved emails
SELECT DISTINCT author
FROM retrieved_emails, authors
WHERE retrieved_emails.author = authors.name;
```

The aggregation of distinct tuples applies here an independence assumption. Alternatively, we could create a disjoint space of retrieved emails, and sum per author over the emails he/she sent.

Next, we estimate the inverse global contribution for each author. Similar to the way we describe the document frequency of a term in the collection, we define the global email contribution of an author. This is simply based on the total number of emails the person wrote. Then, analogous to idf, we describe the inverse global contribution frequency:

```
INSERT INTO inv_gcf
SELECT *
FROM gcf
ASSUMPTION MAX_IDF;
```

This gives us the discriminativeness per author. The idea behind this is that an author who writes in general few emails, but is frequent for a particular topic is likely to be an expert.

Then, join local and global contribution frequency to obtain a ranking of experts.

```
INSERT INTO experts
SELECT author
FROM lcf, inv_gcf
WHERE lcf.author = inv_gcf.author;
```

Since the author relation contains only the email address, but the submission requires candidate id's, we need to join the email address with the candidate id relation.

Expert Search		
Run identifier	MAP	P@10
qmirex1	0.0941	0.1820
qmirex2	0.0473	0.1100
qmirex3	0.0792	0.1740
qmirex4	**0.0959**	**0.1880**

Table 3: Results for Expert Search Task

```
SELECT candidates.id
FROM experts, candidates
WHERE experts.email = candidates.email;
```

In addition to this lcf-gcf ranking strategy for experts, we experimented with alternative combinations and aggregations of the frequencies.

5.2.2 GCF low, LCF low

In this approach we try to find experts who rarely send emails. If an author does not occur frequently among the retrieved emails (lcf low) and if an author occurs rarely in the collection (gcf low), then this author is likely to be an expert on the topic.

5.2.3 GCF low, LCF max

In this approach, if an author wrote the most relevant email among the retrieved emails (lcf max) and if an author occurs rarely in the collection (gcf low), then this author is likely to be an expert on the topic.

5.3 Results

The results for this task are in table 3. They are fairly low quality compared to the average quality achieved for this task. The track organisers conclude for the expert search that taking into account more evidence than just the emails is crucial and adds significant to the achievable retrieval quality.

From this poor result, our main conclusion is that our high abstraction, and minimal resource and coding strategy failed to achieve respectable retrieval quality for the expert search task. For this more complex search task, more resources appear to have a significant effect on the achievable retrieval quality.

We intend to investigate how many person days it takes to explore the given sub-collections, aggregate the evidence, and formulate an effective retrieval function. Further, there is evidence that the inclusion of document length normalisations and statistics on average frequencies into the retrieval strategies does have a significant effect on the retrieval quality, in particular for complex search tasks involving the aggregation of evidence. Therefore, we intend to include length and average functions into our indexes for making respective retrieval functions efficiently executable in probabilistic SQL.

6 Summary and Conclusions

The enterprise TREC participation was for us a welcome opportunity to investigate for different enterprise search tasks whether our retrieval system framework based on probabilistic SQL can cope with TREC-like data, and with how much resource effort we are able to deliver a TREC result.

The result is that with few lines of code in probabilistic SQL, respectable retrieval quality can be achieved for discussion search and known-item search. We observed that disjointness-based are superior to independence-based strategies; this is the expected result. The independence-based strategies, however, are surprisingly close to the disjointness-based strategies. In a number of project contexts, users asked us explicitly for independence-based strategies, as they wish documents to score documents high even if a document fulfills only one "good" criteria.

For expert search, we achieved a poor result. The analysis of the track organisers shows that for this task, the good performers considered more evidence than just the email sub-collection. Our analysis is that a minimal resource strategy is appropriate for classical retrieval tasks, but as soon as several collections and evidence aggregation enters the stage, more attention to detail shows a significant improvement of the retrieval quality, and it is our future aim to optimise the resource effort for achieving a respectable result.

The year 2005 has seen the first TREC participation of the QMUL IR group. From the experience gained we can conclude that participations in the future are reasonable, since with our experimental framework, we can explore data and design retrieval strategies now fairly quickly, even for large data volumes, which has been a problem for a while due to the high abstraction and expressiveness of the data models we apply.

Our next steps include strategies for further scaling our SQL-based processing, and to add length normalisation to the current probabilistic SQL formulation of retrieval strategies.

Thanks to the TREC enterprise and TREC organisers for their effort and excellent support.

References

[Bla88] D. C. Blair. An extended relational document retrieval model. *Information Processing and Management*, 24(3):349–371, 1988.

[Cal94] J. P. Callan. Passage-level evidence in document retrieval. In W. Bruce Croft and C. J. van Rijsbergen, editors, *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 302–310, London, et al., 1994. Springer-Verlag.

[Cra81] R.G. Crawford. The relational model in information retrieval. *Journal of the American Society for Information Science*, 32(1):51–64, 1981.

[DM97] Stefan Dessloch and Nelson M. Mattos. Integrating sql databases with content-specific search engines. In *Proceedings of the 23rd International Conference on Very Large Databases (VLDB)*, pages 528–537, 1997.

[DS04] Nilesh N. Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. In Mario A. Nascimento, M. Tamer Özsu, Donald Kossmann, Renée J. Miller, José A. Blakeley, and K. Bernhard Schiefer, editors, *VLDB*, pages 864–875. Morgan Kaufmann, 2004.

[FGR98] N. Fuhr, N. Gövert, and Th. Rölleke. Dolores: A system for logic-based retrieval of multimedia objects. In W. Bruce Croft, Alistair Moffat, C. J. van Rijsbergen, Ross Wilkinson, and Justin Zobel, editors, *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 257–265, New York, 1998. ACM.

[FR97] N. Fuhr and T. Rölleke. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Transactions on Information Systems*, 14(1):32–66, 1997.

[GF98] David A. Grossman and Ophir Frieder. *Information Retrieval: Algorithms and Heuristics.* Kluwer, Massachusetts, 1998.

[Haw04] David Hawking. Challenges in enterprise search. In *Proceedings of the Australasian Database Conference ADC2004*, pages 15–26, Dunedin, New Zealand, January 2004. Invited paper: url-http://es.csiro.au/pubs/hawking_adc04keynote.pdf.

[Hie03] D. Hiemstra. A database approach to content-based xml retrieval. In *INEX 2002 Workshop Proceedings*, Schloss Dagstuhl, 2003.

[LCIS05] Chengkai Li, Kevin Chen-Chuan Chang, Ihab F. Ilyas, and Sumin Song. Ranksql: query algebra and optimization for relational top-k queries. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 131–142, New York, NY, USA, 2005. ACM Press.

[Roe03] T. Roelleke. The relational Bayes for frequency-based and information-theoretic probability estimation in a probabilistic relational algebra, 2003. Patent application 0322328.6.

[RR02] John Grant Robert Ross, V. S. Subrahmanian. Probabilistic aggregates. In *13th International Symposium on Methodologies for Intelligent Systems (ISMIS), Lyon, France*, Foundations of Intelligent Systems. Springer, 2002.