Predicting the cost-quality trade-off for information retrieval queries: Facilitating database design and query optimization

Henk Ernst Blok
University of Twente
blok@acm.org

Djoerd Hiemstra
University of Twente
hiemstra@cs.utwente.nl

Sunil Choenni
University Nyenrode
s.choenni@nyenrode.nl

Franciska de Jong
University of Twente
fdejong@cs.utwente.nl

Henk M. Blanken
University of Twente
blanken@cs.utwente.nl

Peter M.G. Apers
University of Twente
apers@cs.utwente.nl

ABSTRACT

Efficient, flexible, and scalable integration of full text information retrieval (IR) in a DBMS is not a trivial case. This holds in particular for query optimization in such a context. To facilitate the bulk-oriented behavior of database query processing, a priori knowledge of how to limit the data efficiently prior to query evaluation is very valuable at optimization time. The usually imprecise nature of IR querying provides an extra opportunity to limit the data by a trade-off with the quality of the answer. In this paper we present a mathematically derived model to predict the quality implications of neglecting information before query execution. In particular we investigate the possibility to predict the retrieval quality for a document collection for which no training information is available, which is usually the case in practice. Instead, we construct a model that can be trained on other document collections for which the necessary quality information is available, or can be obtained quite easily. We validate our model for several document collections and present the experimental results. These results show that our model performs quite well, even for the case were we did not train it on the test collection itself.

Keywords

quality, efficiency, trade-off, fragmentation, Zipf, information retrieval, databases

1. INTRODUCTION

In the field of information retrieval (IR), not only the need exists for a system with high precision and recall values, i.e. high retrieval quality (concerning the information need of a user), but also with a low response time. However, in general, the less time we spend on analyzing our document collections, the worse our precision and recall values will be. The past years database research has shown interest in applying database technology in new domains. The seamless integration of IR functionality in a DBMS is one of these new directions. Though integration of IR in a DBMS obviously has several advantages, certain problems still need to be solved. Query optimization, i.e. finding the most efficient way to process a query, is one of these areas. The work presented in this paper is positioned in this area.

In the field of database technology, trade-offs between different parameters, such as query optimization cost and response time, is a cornerstone of research [27]. For example, query optimizers do not search for an optimal query plan but for a good query plan that can be found quickly.

We take a 'database approach' to information retrieval [17, 11, 12, 10, 15], i.e., we do not tie our retrieval model onto a physical data structure, but specify a retrieval model in a declarative way. The information need of a user is represented as an IR query (using natural language), which consists of a set of keywords entered by a user. In our system, this keyword set is modeled as a unary database relation, which is used as a parameter in a database query expression that represents the actual retrieval request. Terms extracted from documents to be indexed are represented as a binary relation `COLL(doc, term)`, in which `term` is a term that appears in document `doc`. At database construction time, two relations with statistics are derived from this relation: `TF(doc, term, tf)`, containing the term frequency per unique document-term pair, and `DF(term, df)`, containing the document frequency per term.

To answer an IR query, tuples of the `TF` and `DF` relations are evaluated. Evaluating the relations completely slows down responsiveness. Neglecting parts of the `DF` relation, and correspondingly of the `TF` relation, will improve the response time but will decrease the quality of the output of a query. Given the imprecise nature of IR queries, insight in the trade-off between response time and quality is useful in building IR systems but also for the user. The user can set threshold values for these parameters, which is a way to express what quality or response time is acceptable. For example, on the one hand, a user may tell an IR system that

the user wants an answer within x time units for a query. The system in its turn can tell the user what quality the user may expect if the response time should be within x time units. On the other hand, the user may also specify to the system what quality is acceptable for him/her.

This paper is devoted to the quantitative aspects of the trade-off between response time and output quality in the context of information retrieval. We present the results how to extrapolate the properties of the trade-off obtained on a certain data set, to another data set which was not involved in the 'training' of the trade-off parameters.

Our approach is based on the following three principles. Firstly, we are interested in top-N queries, a very common form of IR queries. Therefore, we adopt the precision as the basis for our quality notion. To be more precise, we use the average precision, since this metric also includes positional information and not just whether a certain fraction of the top-N is relevant. We do not use recall as our quality metric, since it is too much dependent on N. Note that the recall can always be increased up to 100% by choosing N large enough. Secondly, we choose the terms with highest document frequencies to be the part we neglect during query processing, instead of just a random subset of the data. These terms are considered to be the least discriminative and also provide the numerically least significant per-term contribution to the score of a document in our score function. This means that these terms usually have the least significant impact on the final ranking of an arbitrary document. Furthermore, these terms occupy the most space in the `TF` relation, so ignoring these terms delivers the largest cost reduction. To be more precise, our model uses the fraction of terms with lower document frequencies (i.e., the terms that are not ignored) as its main steering variable. We refer to [2, 1] for a more elaborate description of the underlying fragmenting approach. And, thirdly, we do not try to estimate the quality behavior for each particular query but the quality behavior of the system in general. Therefore, we based our model on the mean of the average precision over a set of queries, instead of just the average precision for a single query.

Given these three choices, we first constructed a model for the simple case. This model can be trained, using the least mean square (LMS) method, on the same collection as one wants to predict the quality for. We call this model, the *first order model*. Next, we present a generalized, *second order*, model that uses the LMS method on the trained parameters of the first order model of a set of training collection to estimate the parameters for the first order model of the test collection. This estimated first order model allows us to predict the quality behavior for that test collection.

This brings us to the main reason for our interest in the cost-quality trade-off in general and its application in integrating IR seamlessly into a database environment [17, 11, 12, 10, 15]. In database query processing, one prefers to work set-at-a-time instead of element-at-a-time. Therefore, traditional IR top-N optimization [23, 5, 4, 9] techniques are sometimes not easy to incorporate in a DBMS [2]. The ability to predict during query optimization which parts of the data set can be ignored — given the knowledge of the related quality implications — would therefore be very interesting. In practice one usually has no significant quality information available for the entire data set, say the document collection statistics in a web search engine. However, for a small (sub)set one might be able to get good relevance judgments given a certain set of representative queries (for instance: the Web TREC plus accompanying queries and relevance judgments). So, it would be useful if one could use this well-documented data set to train a quality model and then transfer the obtained properties to the general case. Our second order model provides precisely these generalization properties.

Furthermore, we are not just interested in transparently integrating an IR system in a DBMS, but in a parallel main-memory DBMS, to investigate the presumed advantages of such a system for IR, and multi-media retrieval in general. Main-memory processing requires the data to be fragmented in advance at data base design time. Therefore, we impose the even higher restriction that we want to be able to model the cost-quality trade-off without detailed knowledge of the actual data. We assume the availability of only limited information, such as the cardinality of certain relations in our database.

In the IR-field, some research has been done on the retrieval cost-effectiveness trade-off [16, 7, 22]. However, we go somewhat further by generalizing quality properties obtained on a given set of collections to another collection, as described above. In database research, the area of probabilistic top-N query optimization [13] closely resembles the basic idea presented in this paper, with the restriction that in that research only the optimization and query evaluation algorithm are probabilistic in a certain sense but the answers are still deterministic. As holds for all database query optimization in general, top-N optimization [6, 14, 8] tries to prune the search space as quickly as possible. In probabilistic query optimization, one tries to guess which parts of the search space are highly unlikely to be of importance to the final answer, so one can ignore these parts as soon as possible. In traditional database probabilistic top-N optimization, a so called restart of the query processing is required when one detects that the search space has been limited too much. In the IR case, we cannot really detect that the search space has been limited too much, since the absolute correctness of the outcome of a query is not defined. We can only try to estimate what the quality implications are of limiting the search space.

2. DEFINITIONS

In this section, we introduce a whole series of mathematical definitions to allow better formalization of the approach later on.

Figure 2 contains the definitions of the set of collections $\mathcal{C}$, denoted by their respective names. To facilitate the discussion of the training and testing of our model in the remainder of this paper we also introduce the notion of a $\mathcal{C}^{train}$ and a $\mathcal{C}^{test}$. How these subsets are constructed is described below.

Since we are interested in fragmenting our database we introduce the fragmenting coefficient f. We limit the possible values of f to those in F, though in theory one could take any $0 \leq f \leq 1$. For convenience sake, we introduce the following terminology:

DEFINITION 1. *An f fragmented collection c is a collection for which only the $f \cdot 100\%$ of the terms with lowest document frequency are taken into consideration during query evaluation (i.e. ranking).*

For each collection $c \in \mathcal{C}$ (f fragmented, where applicable), we define several statistics and data structures (Figure

$$\mathcal{C} \equiv \{\text{FR94, CR, FBIS, FT, LATIMES}\},\ c \in \mathcal{C} \quad (1)$$
$$\mathcal{C}^{train} \subset \mathcal{C}, \text{ a set of training collections} \quad (2)$$
$$\mathcal{C}^{test} \subset \mathcal{C}, \text{ a set of test collections} \quad (3)$$
$$F \equiv \{0.9, 0.925, 0.94, 0.95, 0.96, 0.97, 0.98, 0.985, 0.99, 0.995, 0.999\} \quad (4)$$
$$f \in F \quad (5)$$

Figure 1: Dataset and fragmenting definitions

$$t_{ci} \equiv \text{a term in collection } c \quad (6)$$
$$T_c \equiv \{t_{c1}, t_{c2}, t_{c3}, \ldots,\ t_{ci}, \ldots, t_{n_c}\},\ n_c \equiv |T_c| \quad (7)$$
$$d_{cj} \equiv \text{a document in collection } c,\ d_{cj} \subset T_c \quad (8)$$
$$D_c \equiv \{d_{c1}, d_{c2}, d_{c3}, \ldots,\ d_{cj}, \ldots, d_{c|D_c|}\} \quad (9)$$
$$T_{fc} \equiv \{t_{c1}, t_{c2}, t_{c3}, \ldots,\ t_{ci}, \ldots, t_{cm_c}\} \subset T_c, \text{ where } m_c \equiv f \cdot n_c \equiv |T_{fc}| \quad (10)$$
$$tf_{cij} \equiv \text{term frequency for term } t_{ci} \text{ in document } d_{cj} \quad (11)$$
$$TF_c \equiv \{tf_{cij} | i \in \{1, \ldots, n_c\}, j \in \{1, \ldots, |D_c|\}\} \quad (12)$$
$$ntf_{cij} \equiv \frac{tf_{cij}}{\sum_{i=1}^{n_c} tf_{cij}} \quad (13)$$
$$df_{ci} \equiv |\{(t_{ci}, d) | t_{ci} \in d \in D_c\}| \quad (14)$$
$$DF_c \equiv \{df_{ci} | \forall t_{ci} \in T_c\} \quad (15)$$
$$ndf_{ci} \equiv \frac{df_{ci}}{\tau_c}, \text{ where } \tau_c \equiv \sum_{i=1}^{n_c} df_{ci} \quad (16)$$

Figure 2: Collection statistic definitions

2) We distinguish n_c unique terms after stemming and stopping (Expr. 6 and 7). Note, that we have numbered the terms on ascending document frequency, which we can do without loss of generality (also see Expr. 14). In an f fragmented collection c, we only use the $m_c = fn_c$ first terms (Expr. 10). We model documents as the set of their unique terms (Expr. 8 and 9). For each unique document term pair, we administer the number of times that that term occurs in that particular document (Expr. 11 and 12). We also compute a normalized version (Expr. 13) within the $[0, 1]$ range to facilitate a mathematically better founded score function (see below). For each term, we store the number of documents it occurs in (Expr. 14 and 15). As for the term frequency, we also introduce a normalized version of the document frequency (Expr. 16). As mentioned, we numbered the terms on ascending document frequency, so: $df_{c\,i+1} \geq df_{c\,i}$, $\forall i \in \{1, \ldots, n_c - 1\}$. Next to data sets we also need queries (Figure 2).

$$Q \subset T_c \quad (17)$$
$$\mathcal{Q} \equiv \{Q \in \mathcal{Q}\}, \text{ the set of TREC topics/queries} \quad (18)$$
$$\mathcal{Q}^{train} \subset \mathcal{Q}, \text{ the set of training queries} \quad (19)$$
$$\mathcal{Q}^{test} \subset \mathcal{Q}, \text{ the set of test queries} \quad (20)$$

Figure 3: Query related definitions

For convenience sake, we model our queries as sets, though in reality the queries can contain multiple occurrences of the same term. However, not modeling the queries with this capability will not be a problem given our context. Again, we already distinguish the notion of training and test queries but defer the actual construction of these query sets to the experimental sections below.

Since we need some sub expressions of the score function used in our system (sometimes also known as ranking function), we have listed the relevant definitions in Figure 2.

$$s_{cij} \equiv 1 + \frac{ntf_{cij}}{ndf_{ci}} \quad (21)$$
$$s_{cj}(Q) \equiv \sum_{i \in V} \log(s_{cij}),\ \text{where } V = \{i | t_{ci} \in Q\} \quad (22)$$
$$s_{cfj}(Q) \equiv \sum_{i \in V} \log(s_{cij}),\ \text{where } V = \{i | t_{ci} \in Q \wedge i \leq m_c\} \quad (23)$$
$$p_{ci} \equiv ndf_{ci} \quad (24)$$
$$Es_{cij} \equiv \sum_{i=1}^{n_c} s_{cij} \cdot p_{ci} \quad (25)$$
$$Es_{cfij} \equiv \sum_{i=1}^{m_c} s_{cij} \cdot p_{ci} \quad (26)$$
$$nEs_{cfij} \equiv \frac{Es_{cfij}}{Es_{cij}} \quad (27)$$

Figure 4: Ranking related definitions

Expression 21 is the score contribution of a term i for a document j, as used by our system to rank the documents. The score contribution is motivated by the use of language models for information retrieval, a recently developed approach to information retrieval that performs among the best approaches in experimental evaluations [24, 21]. Expression 22 defines the score of a document given a query Q by summing the logarithm of s_{cij} for each i. The resulting algorithm is a member of the family of $tf \cdot idf$ term weighting algorithms, which are used in many approaches to ranked retrieval [25]. For the relation between language modeling algorithms and the traditional $tf \cdot idf$ term weighting algorithms, we refer to [20]. Expression 23 the variant of the score function used in a fragmented case. For several reasons we need a notion of the probability that a certain term is term i (Expression 24). The main reason to choose the probability this way follows directly from the Zipfian behavior of natural language [28]. Based on this probability, we can define the estimated values in expression 25 and 26. Expression 27 is a normalized version of expression 26. The use of these expressions is clarified in more detail below.

Finally, Figure 2 shows the quality metrics we use.

$$ap_c(Q) \equiv \text{average precision for query } Q \text{ on collection } c \in \mathcal{C} \quad (28)$$
$$ap_{cf}(Q) \equiv \text{average precision for query } Q \text{ on } f \text{ fragmented collection } c \in \mathcal{C} \quad (29)$$
$$map_c \equiv \frac{\sum_{Q \in \mathcal{Q}} ap_c(Q)}{|\mathcal{Q}|} \quad (30)$$
$$map_{cf} \equiv \frac{\sum_{Q \in \mathcal{Q}} ap_{cf}(Q)}{|\mathcal{Q}|} \quad (31)$$
$$nmap_{cf} \equiv \frac{map_{cf}}{map_c} \quad (32)$$

Figure 5: Quality metric definitions

We base our aggregated quality measure on the average precision. For those less familiar with IR terminology: the average precision measure is a single value that is determined for each query. The measure corresponds with a user who walks down a ranked list of documents and will only stop after the user has found a certain number of relevant documents. The measure is the average of the precision calculated at the rank of each relevant document retrieved. Relevant documents that are not retrieved are assigned a

precision value of zero. For example, if three relevant documents exist in the collection and they are retrieved at rank 4, 9, and 20, the average precision would be computed as $\frac{\frac{1}{4}+\frac{2}{9}+\frac{3}{20}}{3} = 0.21$ [19].

Since we are not interested in the quality behavior of just a query in particular, we aggregate over a set of queries (Expressions 30 and 31). Furthermore, we are only interested in relative changes in this aggregated quality measure, so we use a normalized variant (Expression 32).

Besides these model related definitions, we introduce the denotation $\overline{x}$ for the estimated counterpart of a certain variable or parameter x. The main reason for introducing this denotation is the fact that we estimate several parameters and variables in the remainder of this paper, so we need the proper means to distinguish between the actual parameter/variable and its estimated counterpart. We use this denotation recursively. For example, the estimated value of an (already) estimated value $\overline{x}$ is written as $\overline{\overline{x}}$.

To provide a good notion of the error of the models we also define a relative error measure.

DEFINITION 2. *The relative error $\epsilon_{\overline{x}}$ of an estimated value $\overline{x}$ of a variable or parameter x is defined as: $\epsilon_{\overline{x}} \equiv \frac{\overline{x}-x}{x}$*

The advantage of this metric is that it can be *pushed through* the normalization of a relative entity such as $\overline{nmap}_{cf}$.

3. FIRST ORDER APPROACH

In this section, we construct a model to predict $nmap_{cf}$ for a given collection $\mathcal{C}$ that is f fragmented. We train the model using a given set of queries $\mathcal{Q}^{train}$. Once the model has been trained, we test its accurateness on a set of test queries $\mathcal{Q}^{test}$.

3.1 Model

As stated above, we are interested in predicting $nmap_{cf}$ when we know the value of f. Since we do not have any illusion in solving the general problem of information retrieval, i.e. ranking documents perfectly, our only lead is taking a closer look at our ranking method. Of course, no ranking method is perfect, neither is the one we use. However, since a state of the art ranking performs clearly better than just a random ordering of some documents, it clearly does something in the right direction.

The $ap_{cf}(Q)$ for an arbitrary query Q can only change if, as a consequence of decreasing f, a document that correctly appears in the top-N swaps places with a document that should not have been retrieved. In a more formal manner, let us assume we have two documents j_i and j_2 with document scores s_{cj_1} and s_{cj_2}, respectively, such that $s_{cj_1}(Q) \geq s_{cj_2}(Q)$, for an arbitrary query Q. The problem of interest then boils down to the question what should happen to f to make $s_{cfj_1}(Q) < s_{cfj_2}(Q)$, for that same query Q. Trying to come up with an analytical solution for this question appears to be very difficult. However, the main player in the ranking is the score contribution of each individual term i for an arbitrary document j, s_{cij}. The remainder of this paper demonstrates that taking s_{cij} as the basis for estimating $nmap_{cf}$ works out quite well, and in contrast with using s_{cfj} is analytically manageable.

Since we are not interested in actual s_{cij} values but only in its general behavior for 'average' queries and how it degrades for decreasing f we use nEs_{cfij} instead. By taking the expected value instead of the actual value, we abstract from the special effects of just a particular query, likewise we take the mean of the $ap_{cf}(Q)$, i.e. map_{cf}. Dividing by Es_{cij} normalizes the range between 0 and 1 abstracting from the actual numerical range. Similarly, we normalize our quality measure as well, resulting in $nmap_{cf}$ as the actual quality measure instead of $ap_{cf}(Q)$.

Now, let us assume any change in nEs_{cfij} proportionally effects $nmap_{cf}$, in other words:

$$\eta_{c,0} + \eta_{c,1} nEs_{cfij} = nmap_{cf} \quad (33)$$

This leaves us with the question what the influence of f is on nEs_{cfij}. The remainder of this subsection concerns the actual construction of this model for $nmap_{cf}$ with f as explaining variable.

We start with assuming that the df_{ci} values are distributed according to Zipf [28]. We also assumed that the terms are ordered ascendingly on their frequency, so: $df_{ci} \leq df_{ci+1}$. The 'official' Zipf's law assumes a descending order on frequency. Combining this information into one formula gives: $df_{ci} \equiv \frac{a_c}{n_c - i + 1}$ where a_c is the 'constant' in Zipf's law, for a certain collection c. Using expression this formula and the fact that a sum over many small steps can be approximated by an integral we can now rewrite definition 26:

$$\begin{aligned} Es_{cfij} &= \sum_{i=1}^{m_c} s_{cij} \cdot p_{ci} \approx \int_{i=1}^{m_c} s_{cij} \cdot ndf_{ci}\, di \\ &= \int_{i=1}^{m_c} \frac{a_c}{(n-i+1)\tau_c} + ntf_{cij}\, di. \end{aligned} \quad (34)$$

Similarly, we can rewrite definition 25:

$$\begin{aligned} Es_{cij} &= \sum_{i=1}^{n_c} s_{cij} \cdot p_{ci} \\ &\approx \int_{i=1}^{n_c} \frac{a_c}{(n-i+1)\tau_c} + ntf_{cij}\, di \end{aligned} \quad (35)$$

Next, we substitute expressions 34 and 35 in definition 27:

$$nEs_{cfij} = \frac{\int_{i=1}^{m_c} \frac{a_c}{(n-i+1)\tau_c} + ntf_{cij}\, di}{\int_{i=1}^{n_c} \frac{a_c}{(n-i+1)\tau_c} + ntf_{cij}\, di}. \quad (36)$$

Since we do not want to do expensive database accesses, we do not know the value of each ntf_{cij}. Furthermore, for our quality model we are only interested in the global change of the document scores, not in the change in scores of any specific document. Therefore, the normalized term frequency ntf_{cij} might be approximated by γ_c, which is the average normalized term frequency of the document-term pairs in the database. Given this assumption expression 36 reduces to:

$$nEs_{cfij} \approx \frac{\int_{i=1}^{m_c} \frac{a_c}{(n-i+1)\tau_c} + \gamma_c\, di}{\int_{i=1}^{n_c} \frac{a_c}{(n-i+1)\tau_c} + \gamma_c\, di} \quad (37)$$

This effectively reduces our variant of $tf \cdot idf$ weighting to a variant of idf weighting, which was motivated by a Zipf-like distribution in [26]. Although we explicitly derive equation 37 from our language modeling ranking algorithm, we hypothesize that the same approximation holds for any term weighting algorithm that includes an idf component. Evaluation of the integral parts in expression 37 and some further

rewriting results in:

$$nEs_{cfij} \approx \frac{\gamma_c\tau_c m_c}{\aleph} - \frac{a_c \log(n_c - m_c + 1)}{\aleph} - \frac{\gamma_c\tau_c}{\aleph} + \frac{a_c \log n_c}{\aleph}$$

where: $\aleph = \gamma_c\tau_c n_c - \gamma_c\tau_c + a_c \log n_c$. Next, we substitute fn_c for m_c and simplify the expression a bit, using the knowledge that n_c is quite large, resulting in:

$$nEs_{cfij} \approx \frac{\gamma_c\tau_c n_c}{\aleph} f - \frac{a_c}{\aleph}\log(1-f) - \frac{\gamma_c\tau_c}{\aleph} \quad (38)$$

Since, $\aleph$ depends mainly on τ_c and n_c, the second term in this sum will have negligible influence, reducing the basic expression to:

$$nEs_{cfij} \approx \frac{\gamma_c\tau_c n_c}{\aleph} f - \frac{\gamma_c\tau_c}{\aleph} = \varphi_{c,0} f + \varphi_{c,1} \quad (39)$$

where: $\varphi_{c,0} = \frac{\gamma_c\tau_c n_c}{\aleph}$ and $\varphi_{c,1} = -\frac{\gamma_c\tau_c}{\aleph}$. For convenience sake, we rewrite expression 39 into the following form:

$$nEs_{cfij} \approx \gamma_c \left(\varphi'_{c,0} f + \varphi'_{c,1}\right) \quad (40)$$

where:

$$\varphi'_{c,0} = \frac{\tau_c n_c}{\aleph}, \quad (41)$$

$$\varphi'_{c,1} = -\frac{\tau_c}{\aleph} \quad (42)$$

Substituting this rewritten form into expression 33, our relation of interest between nEs_{cfij} and $nmap_{cf}$, gives:

$$\begin{aligned} &\eta_{c,0} + \eta_{c,1} nEs_{cfij} = nmap_{cf} \\ &\Rightarrow \quad \eta_{c,0} + \eta_{c,1}\gamma_c \left(\varphi'_{c,0} f + \varphi'_{c,1}\right) \approx nmap_{cf} \\ &\Rightarrow \quad \psi_{c,0} + \psi_{c,1} f \approx nmap_{cf} \end{aligned} \quad (43)$$

where:

$$\psi_{c,0} = \eta_{c,0} + \eta_{c,1}\gamma_c\varphi'_{c,1} \quad (44)$$

$$\psi_{c,1} = \eta_{c,1}\gamma_c\varphi'_{c,0} \quad (45)$$

Note that expression 43 nicely fits the general observation that the less terms (lower f) one takes into account, the lower the answer quality (lower $nmap_{cf}$) one should expect.

3.2 Experimental setup

Since our model (expression 43) is linear, we can use the LMS (least mean squares) method to estimate the coefficients $\psi_{c,0}$ and $\psi_{c,1}$.

The queries we used, are the 50 retrieval queries, also known as topics in the IR field, from TREC-6. These queries range in length from 9 up to and including 61 terms with an average of 27 terms. This query length might seem unrealistically large since queries typically entered by a user consist of only a few terms. Note however, that many applications exist in which the user does not enter the actual query that is actually executed by the retrieval system. Examples of such situations are: automatic query expansion (used by certain relevance feedback and thesaurus exploiting techniques), or searching for similar texts given an example text (one might think of patent verification).

Since we need a training set $\mathcal{Q}^{train}$ and a test set $\mathcal{Q}^{test}$ of queries, we constructed two random subsets of $\mathcal{Q}$. Both subsets were constructed independently from each other by picking a query from one of the 50 queries in $\mathcal{Q}$ with a uniform probability of 60%. This, of course, does result in some overlap between $\mathcal{Q}^{train}$ and $\mathcal{Q}^{test}$, but since queries are drawn from the same 'virtual' pool in real world application we think this does not inflict a negative impact on the quality of the results. Also, 50 queries as total query pool is not very large so the larger the training and test sets, the less statistical noise we get on the resulting $nmap$, which is averaged over these sets.

The experimental training procedure we followed for each collection $c \in \mathcal{C}$ is described in Figure 3.2.

Step 1 Produce a ranked top-1000 for each query $Q \in \mathcal{Q}^{train}$ on collection c in the normal manner (i.e. by taking all terms into account).

Step 2 Compute the average precision $ap_c(Q)$ for each of the queries of the previous step and compute map_c based on the $ap_c(Q)$ values, according to definition 30.

Step 3 Produce a ranked top-1000 for each query $Q \in \mathcal{Q}^{train}$ on collection c for each $f \in F$.

Step 4 Compute the average precision $ap_{cf}(Q)$ corresponding to each of the results of the previous step and compute map_{cf} based on the $ap_{cf}(Q)$ values, according to definition 31.

Step 5 Compute $nmap_{cf}$ using definition 32.

Step 6 Compute $\overline{\psi}_{c,0}$ and $\overline{\psi}_{c,1}$ using the LMS method on expression 43, given the computed $nmap_{cf}$.

Figure 6: 1rst order training procedure (for a given collection $c \in \mathcal{C}$)

After training our model on a per-collection basis (i.e. determining $\overline{\psi}_{c,0}$ and $\overline{\psi}_{c,1}$), we tested the model using the procedure shown in Figure 3.2 (again, for each collection $c \in \mathcal{C}$).

Step 1 Produce a ranked top-1000 for each query $Q \in \mathcal{Q}^{test}$ on collection c in the normal manner (i.e. by taking all terms into account).

Step 2 Compute the average precision $ap_c(Q)$ for each of the queries of the previous step and compute map_c based on the $ap_c(Q)$ values, according to definition 30.

Step 3 Produce a ranked top-1000 for each query $Q \in \mathcal{Q}^{test}$ on collection c for each $f \in F$.

Step 4 Compute the average precision $ap_{cf}(Q)$ corresponding to each of the results of the previous step and compute map_{cf} based on the $ap_{cf}(Q)$ values, according to definition 31.

Step 5 Compute $nmap_{cf}$ using definition 32.

Step 6 Compute $\overline{nmap}_{cf}$ for each $f \in F$ using expression 43, given $\overline{\psi}_{c,0}$ and $\overline{\psi}_{c,1}$.

Step 7 Compute the relative error between $\overline{nmap}_{cf}$ and $nmap_{cf}$ values using Definition 2.

Figure 7: 1rst order test procedure (for a given collection $c \in \mathcal{C}$)

For the interested reader, we have listed some key statistics of the used document collections in the Appendix, including the actual map_c values.

3.3 Experimental results

In Figure 3.3, we plotted the estimated $nmap_{cf}$ (i.e. $\overline{nmap}_{cf}$) versus the measured $nmap_{cf}$. If our model were perfect, the estimated values would be equal to their corresponding measured value ($\overline{nmap}_{cf} = nmap_{cf}$). We also included this ideal line in the plot for reference.

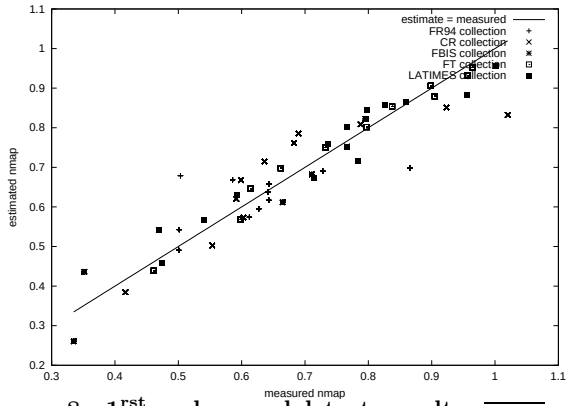

Figure 8: 1rst order model test results, $\overline{nmap}_{cf}$ vs. $nmap_{cf}$, for all collections $c \in \mathcal{C}$

As one can see, all the data points are nicely grouped along the ideal line for all collections. This observation is supported by the relative errors plotted in Figure 3.3.

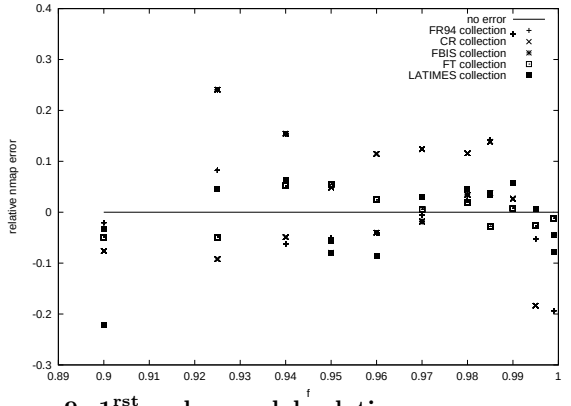

Figure 9: 1rst order model relative error, $\epsilon_{\overline{nmap}_{cf}}$ vs. f, for all collections $c \in \mathcal{C}$

Note, both the values on the axes in Figure 3.3 are relative entities. This means they can range from 0 to 1, and in practice actually do occupy this entire range indeed. Theoretically this is also the case for the *ap* however in practice the *ap* hardly ever exceeds 0.4 for typical $tf \cdot idf$ based retrieval systems. Also note that a relative error $\epsilon_{\overline{nmap}_{cf}} = e$ corresponds to a relative error $\epsilon_{\overline{map}_{cf}} = e$.

4. SECOND ORDER APPROACH

In the previous section, we trained the model parameters on the same collection we wanted to predict the *nmap* for. In reality one wants to predict the *nmap* for a collection for which one has no relevance judgments available. This means that it is impossible to measure the *nmap* in such a case, taking away the possibility to train on the same collection as one wants to predict the *nmap* for.

In other words, one would like to train the model parameters on some collection (or even collections) for which the *nmap* can be measured, and then use this trained model on another collection to predict *nmap* values.

In this section, we use two extra parameters, which also follow nicely from our theoretical model. These two parameters are used to add a second regression model to predict the model parameters of the previous model. At a first glance, this might seem very awkward, but we demonstrate that it is in fact a quite intuitive approach that also provides quite reasonable results.

4.1 Model

In a real world situation it is usually impossible to train the first order model on the same collection c as one wants to predict the quality for, due to the absence of $nmap_c$ and $nmap_{cf}$ information on that collection c. So, one would like to train the model on some other collection $c' \neq c$.

Unfortunately, expression 43 in Section 3 appears not to work in this case. A closer look learns that the estimated coefficients $\overline{\psi}_{c,0}$ and $\overline{\psi}_{c,1}$ differ significantly per collection. It appears that these coefficients contain collection dependent information, as is quite obvious from expression 44 and 45.

In these two parameter definitions, we do not know γ_c, $\eta_{c,0}$, and $\eta_{c,1}$. Also, we do not know $\varphi'_{c,0}$ and $\varphi'_{c,1}$ exactly. But, we know that $\varphi'_{c,0}$ and $\varphi'_{c,1}$ are mainly determined by n_c and τ_c (see expressions 41 and 42), so we approximate the formulas 44 and 45 by:

$$\omega_{0,0} + \omega_{0,1} \cdot \tau_c + \omega_{0,2} \cdot n_c = \psi_{c,0} \quad (46)$$

$$\omega_{1,0} + \omega_{1,1} \cdot \tau_c + \omega_{1,2} \cdot n_c = \psi_{c,1} \quad (47)$$

This completes the construction of our second order model, which clearly captures the collection dependencies in $\psi_{c,0}$ and $\psi_{c,1}$ (and therefore in $\overline{\psi}_{c,0}$ and $\overline{\psi}_{c,1}$).

Note that the expressions 46 and 47 are linear whereas the expressions 41 and 42 are not. We acknowledge that this might be a very crude approximation. The main reason to choose this approximation is the practical advantage in the estimation of the parameters, since we can use the LMS method. The use of two regression models in a nested manner is not an uncommon statistical technique given the type of situation we use it in (see [18]). However, as approximation this model, of course, has to prove its usefulness in practice.

Recall from the definitions in Section 2 that $\tau_c = |TF_c|$ and $n_c = |T_c|$, meaning we only need to perform count operations to get this information. Since this requires practically no database accesses this fits in our requirement to use this model during database design and query optimization. Using these two cheap collection statistics, we can use the expressions 46 and 47, once their own coefficients have been determined, to estimate $\overline{\psi}_{c,0}$ and $\overline{\psi}_{c,1}$.

In the remainder of this section we will demonstrate that the the two expressions 46 and 47 indeed do allow training of our model on collections $c \in \mathcal{C}^{train}$ such that $\mathcal{C}^{train} \cap \mathcal{C}^{test} = \emptyset$.

4.2 Experimental setup

To evaluate our second order approach we extend our first order experimental setup as described in Subsection 3.2. We now also split up our set of collections $\mathcal{C}$ in a set of training collections $\mathcal{C}^{train}$ and test collections $\mathcal{C}^{test}$.

Due to the number of parameters to be estimated in our second order model, we need that $|\mathcal{C}^{train}| \geq 3$. Otherwise, the system is underdetermined. For the special case $|\mathcal{C}^{train}| = 3$ the problem reduces to a system of three equations with three variables, which either has a unique deterministic solution or no solution at all. Since this latter case might cause trouble, though chances that this will happen are very low and the LMS method is in fact perfectly able to determine the solution if one exists, we require that $|\mathcal{C}^{train}| > 3$.

To allow the most accurate training we looked only at cases where $|\mathcal{C}^{test}| = 1$ leaving 4 collections to train our model on (which we do need anyway, as we just argued). Consequently, we have 5 possible ways to divide $\mathcal{C}$ in a $\mathcal{C}^{train}$ and $\mathcal{C}^{test}$, by subsequentially taking each $c \in \mathcal{C}$ as test collection ($\mathcal{C}^{test} = \{c\}$) and the remaining 4 as training collections ($\mathcal{C}^{train} = \mathcal{C} - \{c\}$). For a given partitioning of $\mathcal{C}$ in $\mathcal{C}^{train}$ and $\mathcal{C}^{test}$ that way, we can train, and subsequently test, our model. Figure 4.2 describes the training procedure we followed.

Step 1 Perform the first order training procedure (see Figure 3.2, Subsection 3.2) to compute $\overline{\psi}_{c,0}$ and $\overline{\psi}_{c,1}$ for each collection $c \in \mathcal{C}^{train}$.

Step 2 Get τ_c and n_c for each collection $c \in \mathcal{C}^{train}$.

Step 3a Use the results from **Step 1** ($\overline{\psi}_{c,0}$) and **Step 2** (τ_c and n_c) in combination with the LMS method on equation 46 to estimate $\overline{\omega}_{0,0}$, $\overline{\omega}_{0,1}$, and $\overline{\omega}_{0,2}$.

Step 3b Use the results from **Step 1** ($\overline{\psi}_{c,1}$) and **Step 2** (τ_c and n_c) in combination with the LMS method on equation 47 to estimate $\overline{\omega}_{1,0}$, $\overline{\omega}_{1,1}$, and $\overline{\omega}_{1,2}$.

Figure 10: 2nd order training procedure (for a given partitioning of $\mathcal{C}$ in $\mathcal{C}^{train}$ and $\mathcal{C}^{test}$ where $|\mathcal{C}^{test}| = 1$)

Figure 4.2 describes the corresponding test procedure we followed, using the $(\overline{\omega}_{0,0}, \overline{\omega}_{0,1}, \overline{\omega}_{0,2})$ and $(\overline{\omega}_{1,0}, \overline{\omega}_{1,1}, \overline{\omega}_{1,2})$ vectors that were determined according to the procedure described in Figure 4.2.

Step 1 Get τ_c and n_c for the test collection $c \in \mathcal{C}^{test}$.

Step 2a Substitute $\overline{\omega}_{0,0}$, $\overline{\omega}_{0,1}$, $\overline{\omega}_{0,2}$, τ_c, and n_c in expression 46 to compute $\overline{\overline{\psi}}_{c,0}$.

Step 2b Substitute $\overline{\omega}_{1,0}$, $\overline{\omega}_{1,1}$, $\overline{\omega}_{1,2}$, τ_c, and n_c in expression 47 to compute $\overline{\overline{\psi}}_{c,1}$.

Step 3 Perform the first order test procedure (see Figure 3.2, Subsection 3.2) for test collection $c \in \mathcal{C}^{train}$. However, now use $\overline{\overline{\psi}}_{c,0}$ and $\overline{\overline{\psi}}_{c,1}$ in **Step 6** instead of $\overline{\psi}_{c,0}$ and $\overline{\psi}_{c,1}$, respectively.

Figure 11: 2nd order test procedure (for a given partitioning of $\mathcal{C}$ in $\mathcal{C}^{train}$ and $\mathcal{C}^{test}$ where $|\mathcal{C}^{test}| = 1$)

4.3 Experimental results

In Figure 5 we combined all five test cases, each represented by their respective test collection. As before, we plotted the $\overline{nmap}_{cf}$ vs. $nmap_{cf}$. We also included the ideal line where $\overline{nmap}_{cf} = nmap_{cf}$.

As expected the point clouds do not group as nicely along the ideal line as in the first order case (Subsection 3.3). However, as we can see in Figure 5, the relative error stays within 25% in most of the cases. We find this quite acceptable, given the fact we use only very little information in our model (f, τ_c, and n_c). Furthermore, we stress on the fact that the number of training collections is very low from a statistical point of view.

A closer review of the log files of the first and second order training and testing runs also learned us that the number of 50 queries in total, is very low as well (recall that we had to do with these 50 queries for both the training and testing).

5. CONCLUSIONS AND FUTURE WORK

In this paper, we derived a mathematical model to estimate the expected decrease in retrieval answer quality given that we use only the fraction f of the terms with the lowest document frequencies.

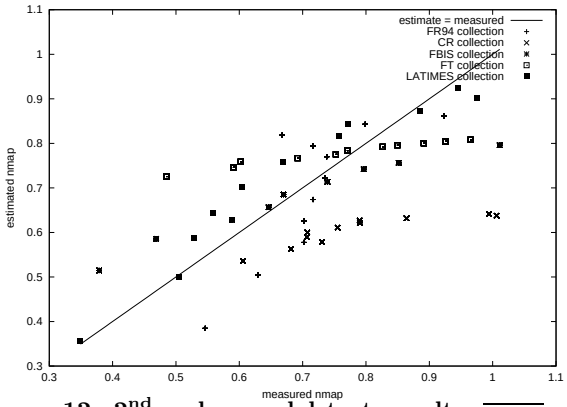

Figure 12: 2nd order model test results, $\overline{nmap}_{cf}$ vs. $nmap_{cf}$, for all collections $c \in \mathcal{C}^{test}$ and all $\mathcal{C}^{test} \subset \mathcal{C}$ where $|\mathcal{C}^{test}| = 1$

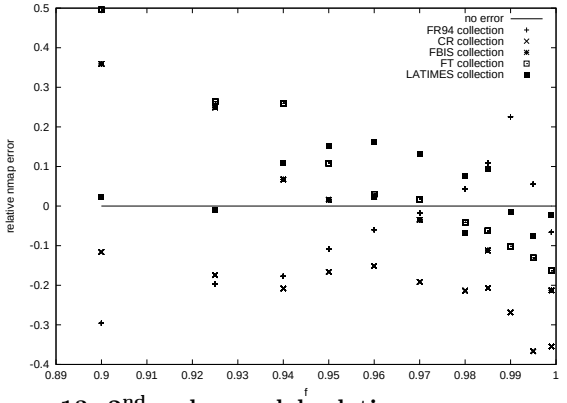

Figure 13: 2nd order model relative error, $\epsilon_{\overline{nmap}_{cf}}$ vs. f, for all collections $c \in \mathcal{C}^{test}$ and all $\mathcal{C}^{test} \subset \mathcal{C}$ where $|\mathcal{C}^{test}| = 1$

We distinguished two major approaches, of which the second was an extension of the first. The performed experiments demonstrated that for the first approach, in which we tested our model on the same collection as we trained its parameters on, our model predicts the quality implications of degrading f very well. The second approach attempted to overcome the major drawback of the first approach. There we trained the model on different collections than we wanted to test it on, i.e. predict the quality for. We observed a significant increase in the (relative) estimation error, which we expected beforehand. However, the results are still interesting, since the relative error stayed mostly within the 25% range.

In the construction of our first order model in Section 3, we assumed any change in nEs_{cfij} to effect $nmap_{cf}$ proportionally (Expression 33). Given the rather good experimental results obtained with our first order model, we have no reason to question this assumption on its practical effectiveness. However, we certain are interested in a more formal

derivation of this relation, so it is certainly a candidate for future research. Likewise, we have no reason to withdraw the approximation of ntf_{cij} by a constant γ_c.

But for the approximation of the formulas 44 and 45 (which in fact imply approximations of the Expressions 41 and 42) by the formulas 46 and 47 we are not so certain. The results of the second order model can be considered quite reasonable, given the fact that we wanted to use only very little information, and that it was a first attempt to predict retrieval quality using a model trained on other collections. However, the results are not that good that we are willing to accept the approximation blindly. Furthermore, we did approximate a non-linear mapping by a simple linear one. Mathematically speaking, such an approximation has a high chance of miss-fitting the original mapping quite a bit. So, we certainly think this approximation should be investigated in more detail in the future.

Furthermore, we want to stress that the statistical stability of our experiments seemed to have suffered somewhat from the lack of data. This holds in particular for the experiments we performed for the second order model. We recommend to repeat these experiments with more and different document collections and more queries. Due to several practical reasons, including the limited space we have in this paper, we are not able to report any results on that here. However, we are working on evaluation of our models on the Web TREC datasets (1, 10 and, 100 GB). The current queries can also be extended with the topics of other TREC conferences (we only used the topics of TREC-6).

Finally, we plan to link our quality prediction model to our cost model that also uses f as one of its main parameters (also see [1]). This integration would indeed provide a direct trade-off between the executions costs and the retrieval quality, which we plan to incorporate into our DBMS.

An extended version of this paper is available as [3].

6. REFERENCES

[1] H. E. Blok, S. Choenni, H. M. Blanken, and P. M. Apers. A selectivity model for fragmented relations in information retrieval. CTIT Tech. Report 01-02, Enschede, The Netherlands, Feb. 2001.

[2] H. E. Blok, A. P. de Vries, H. M. Blanken, and P. M. Apers. Experiences with IR TOP N Optimization in a Main Memory DBMS: Applying 'the Database Approach' in New Domains. In *BNCOD 18*, volume 2097 of *LNCS*. Springer, July 2001.

[3] H. E. Blok, D. Hiemstra, S. Choenni, F. de Jong, H. M. Blanken, and P. M. Apers. Predicting the cost-quality trade-off for information retrieval queries: Facilitating database design and query optimization. CTIT Tech. Report, Enschede, The Netherlands, 2001.

[4] E. W. Brown. Execution Performance Issues in Full-Text Information Retrieval. Ph.D. Thesis/Technical Report 95-81, University of Massachusetts, Amherst, Oct. 1995.

[5] C. Buckley and A. F. Lewit. Optimisation of Inverted Vector Searches. In *ACM SIGIR Conference*, pages 97–110, Aug. 1985.

[6] M. J. Carey and D. Kossmann. Reducing the Braking Distance of an SQL Query Engine. In *VLDB Conference*, pages 158–169, 1998.

[7] U. Çetintemel, B. T. Jónsson, M. J. Franklin, C. L. Giles, and D. Srivastava. Evaluating Answer Quality/Efficiency Tradeoffs. In *KRDB Workshop*, volume 10 of *CEUR Workshop Proceedings*, pages 19.1–19.4, May 1998.

[8] S. Chaudhuri and L. Gravano. Evaluating Top-k Selection Queries. In *VLDB Conference*, pages 397–410, 1999.

[9] D. R. Cutting and J. O. Pedersen. Space Optimizations for Total Ranking. In *RAIO Conference*, pages 401–412, June 1997.

[10] A. P. de Vries. *Content and Multimedia Database Management Systems*. PhD thesis, University of Twente, Enschede, The Netherlands, Dec. 1999.

[11] A. P. de Vries and H. M. Blanken. The Relationship between IR and Multimedia Databases. In *IRSG Colloquium on Information Retrieval*, 1998.

[12] A. P. de Vries and A. N. Wilschut. On the Integration of IR and Databases. In *8th IFIP 2.6 Working Conference on Data Semantics 8*, 1999.

[13] D. Donjerkovic and R. Ramakrishnan. Probabilistic Optimization of Top N Queries. In *VLDB Conference*, pages 411–422, 1999.

[14] M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, and J. D. Ullman. Computing Iceberg Queries Efficiently. In *VLDB Conference*, pages 299–310, 1998.

[15] O. Frieder, D. Grossman, A. Chowdhury, and G. Frieder. Efficiency Considerations for Scalable Information Retrieval Servers. *Journal of Digital Information*, 1(5), 2000.

[16] D. Grossman, D. Holmes, and O. Frieder. A Parallel DBMS Approach to IR in TREC-3. In *TREC-3 Conference*, NIST Special publications, pages 279–288, Nov. 1995.

[17] D. A. Grossman and O. Frieder. *Information retrieval: algorithms and heuristics*. Kluwer Academic, Boston, 1998. ISBN 0-7923-8271-4.

[18] J. Hair, R. Anderson, R. Tatham, and W. Black. *Mutlivariate Data Analysis*. Prentice Hall, Inc, New Jersey, 5th edition, 1998. ISBN 0-13-930587-4.

[19] D. Harman. Evaluation Techniques and Measures. In *TREC-3 Conference*, NIST Special publications, pages A5–A13, Nov. 1995.

[20] D. Hiemstra. A probabilistic justification for using $tf \cdot idf$ term weighting in information retrieval. *International Journal on Digital Libraries*, 3(2):131–139, 2000.

[21] D. Hiemstra and W. Kraaij. Twenty-One at TREC-7: Ad-hoc and Cross-language track. In *TREC-7 Conference*, NIST Special publications, pages 227–238, Nov. 1999.

[22] B. T. Jónsson, M. J. Franklin, and D. Srivastava. Interaction of Query evaluation and Buffer management for IR. In *ACM SIGMOD Conference*, 1998.

[23] M. Persin. Document filtering for fast ranking. In *ACM SIGIR Conference*, pages 339–349, Aug. 1984.

[24] J. M. Ponte and W. B. Croft. A Language Modeling Approach to Information Retrieval. In *ACM SIGIR Conference*, pages 275–281, Aug. 1998.

[25] G. Salton and C. Buckley. Term-Weighting Approaches in Automatic Text Retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[26] K. Sparck-Jones. A Statistical Interpretation of Term Specifity and its Application in Retrieval. *Journal of Documentation*, 28(1):11–20, 1972.

[27] J. D. Ullman. *Principles of database and knowledgebase systems*, volume 2 of *Principles of computer science series*. Computer Science Press, Rockville, Maryland, 1988. ISBN 0-7167-8162-X.

[28] G. Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, Reading, MA, USA, 1949.