From Retrieval Status Values to Probabilities of Relevance for Advanced IR Applications

Henrik Nottelmann, Norbert Fuhr

Institute of Informatics and Interactive Systems, University of Duisburg-Essen,
47048 Duisburg, Germany, {nottelmann, fuhr}@uni-duisburg.de

Information Retrieval systems typically sort the result with respect to document retrieval status values (RSV). According to the Probability Ranking Principle, this ranking ensures optimum retrieval quality if the RSVs are monotonously increasing with the probabilities of relevance (as e.g. for probabilistic IR models). However, advanced applications like filtering or distributed retrieval require estimates of the actual probability of relevance. The relationship between the RSV of a document and its probability of relevance can be described by a “normalisation” function which maps the retrieval status value onto the probability of relevance (“mapping functions”). In this paper, we explore the use of linear and logistic mapping functions for different retrieval methods. In a series of upper-bound experiments, we compare the approximation quality of the different mapping functions. We also investigate the effect on the resulting retrieval quality in distributed retrieval (only merging, without resource selection). These experiments show that good estimates of the actual probability of relevance can be achieved, and that the logistic model outperforms the linear one. Retrieval quality for distributed retrieval is only slightly improved by using the logistic function.

1 Introduction

Information Retrieval systems typically rank documents according to their “retrieval status values” with respect to (w. r. t.) the given query. In the Boolean model, RSVs are either zero or one. Fuzzy retrieval allows for RSVs in the interval $[0, 1]$. The well-known vector-space-model can be used with the cosine metric (RSVs in $[0, 1]$) or the scalar product (RSVs in $\mathbb{R}$).

Probabilistic models are widely used in Information Retrieval. Besides the fact that even classical non-probabilistic models can be given a probabilistic interpretation [25], current language models extend classical probabilistic models by methods for considering specific representations of documents and queries. The key advantage of probabilistic models is their underlying theoretic justification. The Probability Ranking Principle (PRP) [20] states that optimum retrieval (defined w. r. t. document representations) is given if the documents are ranked according to the probability $Pr(\text{rel}|d, q)$ that document d is relevant to a user query q (“probability of relevance”).

For ad-hoc retrieval, probabilistic IR algorithms do not have to estimate these probabilities of relevance directly; instead it is sufficient to rank the documents according to the documents’ retrieval status values if these are monotonously increasing with $Pr(\text{rel}|d, q)$. For example, the well-known binary independence retrieval model ranks documents according to $Pr(d|\text{rel}, q)/Pr(d|\neg\text{rel}, q)$. Computation of these RSVs is easier than estimating the exact probabilities of relevance, and it is sufficient for ad-hoc retrieval.

As a consequence, little effort has been spent on approximating the relationship between retrieval status values and probabilities of relevance (specified by a “mapping function”).

For Rijsbergen’s paradigm of uncertain inference (a probabilistic generalisation of the logical view on databases) [23], a linear relationship between the retrieval status value $Pr(q \leftarrow d)$ (“probability of inference”) and the probability of relevance has been proposed [24].

Recently, [13] proposed a mixture model for approximating the RSV distribution. The RSVs of the relevant documents are modelled by a normal (Gaussian) distribution, and the RSVs of the non-relevant documents are approximated by an exponential distribution. These distributions and the mixing parameters can be easily used for computing the conditional probability that a document is relevant if it has a given score. This model, however, cannot be used in practice, as it requires about 60 relevant documents for computing good parameters, which are not available in most cases.

In general, the estimation of the probabilities of relevance based on the document’s RSV allows for comparing different weighting schemes and retrieval methods on the same or different collections; here mapping RSVs onto probabilities of relevance acts as a normalisation of the document scores. In fact, advanced IR applications actually need estimations of the probabilities of relevance:

Filtering: Following the decision-theoretic justification of the PRP, a document should only be presented to the user if the costs for retrieval are lower than those for omitting the document, i.e.

$$Pr(rel|q, d) \cdot C_r + [1 - Pr(rel|q, d)] \cdot \bar{C}_r < Pr(rel|q, d) \cdot C_o + [1 - Pr(rel|q, d)] \cdot \bar{C}_o, \quad (1)$$

where C_r ($\bar{C}_r$) denote the costs for retrieving a relevant (non-relevant) document; similarly, C_o ($\bar{C}_o$) is the cost for omitting a relevant (non-relevant) document from the retrieved set.

Resource selection: In distributed IR, resource selection is the task to determine the best collections to be searched. The decision-theoretic framework [14] for resource selection aims at estimating the number of relevant documents;¹ for this, the probabilities of relevance of the top-ranked documents have to be approximated.

Data fusion: The probabilities of relevance also play an important role in the fusion step of distributed retrieval, where the documents retrieved from the selected collections have to be merged in order to get a single ranked list. Using the probabilities of relevance as a normalisation of the RSVs is a natural and theoretically well-founded solution to this problem.

We already investigated mapping functions for the paradigm of uncertain inference [15]. Here, logistic functions proved to be more effective for approximating the relationship between $Pr(q \leftarrow d)$ (the RSV in uncertain inference) and $Pr(rel|q, d)$ than linear functions. Logistic functions have been used in different application areas within IR for quite some time, e.g. for text categorisation [8] or retrieval functions [4, 9] (logistic variant of the model proposed in [7]).

In this paper, we extend this work by evaluating linear and logistic functions for three different retrieval methods: TF-IDF, INQUERY and language models. For these models, linear functions can be regarded as simple forms of mapping functions although they cannot be derived theoretically (as for uncertain inference). Logistic functions are used as a good approximation of the ideal case, in which a step function has to be used as mapping function.

This paper is organised as follows. Section 2 describes briefly the retrieval methods and the mapping functions we consider. Section 3 reports on the results of our evaluation; we measured the overall quality of the estimates of the real probabilities as well as their impact on retrieval quality in the context of distributed IR. Finally, section 4 contains concluding remarks and an outlook on future work.

2 Retrieval status values and probabilities of relevance

Usually, information retrieval engines compute document weights, called “retrieval status values”(RSV), denoted by $RSV(d, q)$ for document d and query q . Different IR methods compute RSVs in different ways and different scales (e.g. $\{0, 1\}$, $[0, 1]$, $\mathbb{R}$). However, for advanced applications we need the probability $Pr(rel|d, q)$ that d is relevant to q (“probability of relevance”).

¹In contrast to resource-ranking algorithms like GLOSS [11, 10] or CORI [2] which only compute a matching score between collections and the given query.

The actual relationship between the RSV of a document and its probability of relevance is approximated by a “mapping function”

$$f : \mathcal{R} \mapsto [0, 1], \quad f(RSV(d, q)) \approx Pr(\text{rel}|q, d). \quad (2)$$

In the rest of this section, we will describe briefly different retrieval methods for computing the RSV, and the mapping functions we will consider in the evaluation part of this paper.

2.1 Retrieval methods

In this paper, we investigate three different retrieval methods (see below). In the following definitions, $tf(t, d)$ is the term frequency (number of times term t occurs in document d), $dl(d)$ denotes the document length (number of terms in document d), $avgdl$ the average document length, $|DL|$ the collection size (number of documents), and $df(t)$ the document frequency (number of documents containing term t). The weight of term t in document d is denoted by $dtf(t, d)$, and the query term weight by $qtf(t, q)$.

We consider three different retrieval methods, all of them implemented in the Lemur toolkit ²:

TF-IDF: This weighting scheme became popular with the SMART retrieval engine [21]. SMART is based on the vector-space model, where documents and queries are points in a high-dimensional vector space defined by the terms which are considered to be orthogonal. For retrieval, documents whose term vector is similar to the query vector are returned. Similarity in SMART is measured as the cosine of the angle between the vectors.

We use a simple version of TF-IDF which computes the scalar product of the query and the document vector

$$RSV(d, q) := \sum_{t \in q} dtf(t, d) \cdot qtf(t, q) \quad (3)$$

with $tf \cdot tfx$ [22] document weights

$$dtf_{TF-IDF}(t, d) := tf(t, d) \cdot \log \frac{|DL| + 1}{df(t) + 0.5}. \quad (4)$$

As this implementation of TF-IDF does not normalise the term frequency, it is biased towards longer documents.

Raw tf values $tf(t, q)$ are used as query term weights.

INQUERY: The INQUERY system [3, 1] is based on inference networks with document nodes, indexing term nodes, query concept (i.e., query term) nodes and a single query node. The probability of the arcs between a document node and text nodes is set to

$$dtf_{INQUERY}(t, d) := 0.4 + 0.6 \cdot \frac{tf(t, d)}{tf(t, d) + 0.5 + 1.5 \cdot \frac{dl(d)}{avgdl}} \cdot \frac{\log \frac{|DL| + 0.5}{df(t)}}{\log(|DL| + 1)}. \quad (5)$$

This weighting scheme is very similar to the BM25 implementation we used in our previous experiments [15].

In the query network, the nodes of the query terms are connected to the single query node, computing the final score by the linear retrieval function in formula 3.

LM: Statistical language models have a long history in the fields of speech recognition and statistical language processing, and proved recently to be effective for IR [17]. The basic idea is that queries and documents are generated by statistical language models. E.g., the document language model is described by the “indexing weights”

$$dtw_{LM}(t, d) = Pr(t|d) = \frac{tf(t, d)}{dl(d)} \quad (6)$$

²<http://www-2.cs.cmu.edu/~lemur>

(maximum-likelihood estimation). Due to sparse data, this probability is smoothed by a collection-specific language model $Pr(t|DL)$ (Jelinek-Mercer interpolation, typically with $\lambda = 0.5$):

$$Pr_s(t|d) = \lambda Pr(t|d) + (1 - \lambda) Pr(t|DL). \quad (7)$$

In Lemur, documents are then ranked w. r. t. the KL divergence

$$D(q||d) = \sum_{t \in q} Pr_s(t|q) \log \frac{Pr_s(t|q)}{Pr_s(t|d)}. \quad (8)$$

2.2 Mapping functions

As explained above, for some applications we need to map the retrieval status value $RSV(d, q)$ of a document d (w. r. t. query q) onto the probability $Pr(\text{rel}|d, q)$ that it is relevant to a user. These probabilities of relevance are computed by a “mapping function” (see formula 2).

We consider three different functions: Pure linear functions, affine linear functions and logistic functions.

2.2.1 Linear functions

Obviously, the probability of relevance should be monotonously increasing with the retrieval status value (assuming that the retrieval engine provides a proper ranking). A first and very simple approximation of this relationship is a pure linear function, where the probability that a document is relevant is proportional to the retrieval status value:

$$f_{lin}(x) := c_1 \cdot x. \quad (9)$$

We can control the mapping function in a more flexible way by adding a second degree of freedom, arriving at an affine linear function:

$$f_{alin}(x) := c_0 + c_1 \cdot x. \quad (10)$$

Both linear functions have the same drawback. They do not ensure that the results are between 0 and 1 in the general case of $c_0, c_1 \in \mathbb{R}$. In other words, the result cannot necessarily be regarded as a probability.

However, linear mapping functions can be justified in the context of uncertain inference [23]: This IR paradigm is an adoption of the logical view on database, where queries and document contents are treated as logical formulae, and a database only returns those documents d which logically imply the query q , i.e. it proves $q \leftarrow d$. External knowledge like a thesaurus or an ontology can easily be integrated. For considering the intrinsic uncertainty of information retrieval, Rijsbergen interpreted probabilistic IR as estimating the probability $Pr(q \leftarrow d)$ that the document logically implies the query. However, this probability cannot be considered in the traditional sense of logical databases, i.e. $Pr(q \leftarrow d) \not\equiv Pr(\neg d \wedge q)$, but as the conditional probability $Pr(q|d)$.

So far, this model does not cope with the concept of relevance. If we assume a monotonically increasing function mapping the probability of inference onto the probability of relevance, it is sufficient to rank documents according to $Pr(q \leftarrow d)$. Thus, only little work has been done on determining a “good” mapping function.

One of the possible mappings from the probability of inference $Pr(q \leftarrow d)$ onto the probability of relevance $Pr(\text{rel}|q, d)$ is given in [24] based on the total probability theorem:

$$Pr(\text{rel}|q, d) = Pr(\text{rel}|q \leftarrow d) \cdot Pr(q \leftarrow d) + Pr(\text{rel}|\neg(q \leftarrow d)) \cdot Pr(\neg(q \leftarrow d)) \quad (11)$$

$$= Pr(\text{rel}|q \leftarrow d) \cdot Pr(q \leftarrow d) + Pr(\text{rel}|\neg(q \leftarrow d)) \cdot [1 - Pr(q \leftarrow d)] \quad (12)$$

$$= \underbrace{Pr(\text{rel}|\neg(q \leftarrow d))}_{c_0} + \underbrace{[Pr(\text{rel}|q \leftarrow d) - Pr(\text{rel}|\neg(q \leftarrow d))]}_{c_1} \cdot Pr(q \leftarrow d) \quad (13)$$

$$= f(Pr(q \leftarrow d)) . \quad (14)$$

Thus, we arrive at an affine linear mapping function. By assuming $c_0 = Pr(\text{rel}|\neg(q \leftarrow d)) \approx 0$, the formula turns into a pure linear function.

In principal, the conditional probabilities $Pr(\text{rel}|q \leftarrow d)$ and $Pr(\text{rel}|\neg(q \leftarrow d))$ —and, thus, also c_0 and c_1 —depend on the document and query, but cannot be computed in advance. Thus, we will assume that the parameters do not depend on a particular document or query.

2.2.2 Logistic functions

In an ideal situation, exactly the documents in the ranks $1, \dots, l$ are relevant, and the documents in the remaining ranks $l+1, \dots$ are irrelevant. Let a be the RSV of the documents in rank l (i.e., the lowest RSV of any relevant documents). Thus, the relationship function should be a step function

$$f : \mathbb{R} \rightarrow [0, 1], f(x) := \begin{cases} 1 & , \quad x \geq a, \\ 0 & , \quad x < a \end{cases} \quad (15)$$

where x is the document's RSV.

Obviously, no information retrieval system can ensure this requirement; in general some of the top-ranked documents are irrelevant, and some of the lower-ranked documents are relevant. But, less documents with lower RSV should be relevant than documents with higher weights. In other words, the probability that any arbitrary document is relevant should decrease with decreasing RSV.

For modelling this characteristic, we want a continuous function f which approximates the discrete step function (15). Obviously, the pure and affine linear functions (10,9) are not appropriate.

Instead, one good candidate is the logistic function [5, 6]

$$f_{log} : \mathbb{R} \rightarrow [0, 1], f_{log}(x) := \frac{\exp(b_0 + b_1 \cdot x)}{1 + \exp(b_0 + b_1 \cdot x)} \quad (16)$$

with the two parameters b_0 and b_1 . Figure 1 depicts some logistic functions with different parameters. One of the nice properties of logistic functions is that the result is always in $[0, 1]$. In addition, a large variety of curves can be obtained by varying b_0 and b_1 . The curve can be moved along the x axis by varying b_0 ; the slope can be adjusted by varying b_1 .

Figure 1: Example logistic functions

2.2.3 Parameters of the mapping functions

All functions presented so far—linear as well as logistic ones—depend on some query-specific parameters c_0 , c_1 , b_0 and b_1 .

Given example tuples (document, query, RSV, relevance judgement), parameters can be learned by means of regression methods. Possible optimisation criteria are maximum likelihood [8] or least-square

polynomials [16]. In both cases, extrema of a function (the likelihood function or the square error) have to be determined, i.e. the points where the first derivative equals zero. The resulting equation cannot be solved directly; instead, an iterative method (e.g. Newton-Raphson) has to be applied.

Usually the parameters are unknown before retrieval, and relevance feedback data is not available. Therefore, query-independent parameters have to be learned in a separate training phase (with a sample based on several queries). Obviously, the parameters depend on both the collection and the retrieval method; i.e., mapping function parameters have to be learned for each collection and each retrieval method separately.

3 Evaluation

This section presents the experiments we conducted. We first describe the test-bed, including the documents, the queries, the relevance judgements and the learning algorithms. Then, we present retrieval results for the three different retrieval methods without any mapping function (baseline results). In subsection 3.3, we describe briefly how we conduct our experiments. For every retrieval method, we compute the approximation error of each mapping function using optimum parameters. In addition, we investigate retrieval quality in a distributed IR setting without resource selection (using a cross-evaluation method for the parameters). The rest of this section contains the results for TF-IDF, INQUERY and language models (LM).

3.1 Experimental Setup

The work described in this paper originates from work in the field of distributed information retrieval. Thus, we used the TREC-123 test bed with the CMU 100 collection split [2] which is heavily used for evaluating distributed IR methods. The collections are of roughly the same size (about 33 megabytes), but vary in the number of documents they contain. The documents inside a collection are from the same source and the same time-frame. Table 1 depicts the summarised statistics for this 100 library test-bed.

Table 1: Summarised statistics for the 100 collection test-bed

Collection	Minimum	Average	Maximum
Documents	752	10,782	33,723
Bytes	28,070,646	33,365,514	41,796,822

The document index only contains the `<text>` sections of the documents (with the different document indexing weights $d_{tf}(t, d)$ as described in subsection 2.1).

Queries are based on TREC topics 51–100 and 101–150 [12], respectively. We use three different sets of queries:

1. Short queries, where we only used the `<title>` field. Short queries contain between 1 and 7 terms (average 3.3), and are similar to those submitted to a WWW search engine.
2. Mid-length queries, where we only used the `<description>` field. Here, queries typically contain between 4 and 19 terms (average 9.9), and may be used by advanced searchers.
3. Long queries, where we used all fields. These queries contain 39–185 terms (average 87.5) and are common in TREC-based evaluations.

For both documents and queries, terms are stemmed (using the Porter stemmer [18]), and stop words (the TREC “common words”) are removed.

The relevance judgements are the standard TREC relevance judgements [12], documents with no judgement are treated as irrelevant.

In our experiments, we use the Gnuplot³ implementation of the nonlinear least-squares (NLLS) Marquardt-Levenberg algorithm [19] for learning the parameters c_0, c_1 of the linear function and b_0, b_1 of the logistic function.

Retrieval for all three retrieval methods (TF-IDF, INQUERY, LM) is carried out with the Lemur toolkit.

3.2 Baseline experiments

In a first set of experiments, we compared the retrieval quality (w.r.t. the RSVs) of the three different retrieval methods. Thus, we can test whether specific mapping functions are more suitable for high-performance retrieval methods and others for poor performing methods.

For these baseline experiments, we merged all 100 collections into one big collection and ran our 100 topics (51–150) on this collection without considering mapping functions (i.e. the documents are ranked w. r. t. their RSVs).

For TF-IDF, quality is bad on the whole collection (though it is quite good on single collections). The problem is that the implementation we use does not normalise term frequencies, thus ranking longer documents high although they might not be relevant. As a consequence, nearly all of the top-ranked documents are from FR (Federal Register) or PT (Patents). These collections contain fewer but larger documents than the “average” collection, but hardly any relevant documents for many queries. Thus we left out FR and PT for this baseline experiment.

The results shown in Tab. 2 and Fig. 2 are clear: INQUERY performs good, where TF-IDF is still extraordinary bad (even on TREC-123 without FR and PT). LM (language models) perform worse than INQUERY for short and mid-length queries; for long queries, their retrieval quality is as good as for INQUERY.

Not surprisingly, effectiveness improves when queries become larger.

Table 2: Baseline experiments: Average precision at given ranks

Rank	short queries			mid-length queries			long queries		
	TF-IDF	INQ.	LM	TF-IDF	INQ.	LM	TF-IDF	INQ.	LM
5	0.2280	0.4600	0.3600	0.0460	0.4540	0.3880	0.1940	0.6340	0.6500
10	0.2300	0.4540	0.3790	0.0510	0.4540	0.3910	0.1830	0.6370	0.6370
15	0.2267	0.4473	0.3827	0.0553	0.4480	0.3833	0.1740	0.6253	0.6133
20	0.2215	0.4345	0.3765	0.0540	0.4345	0.3770	0.1710	0.6080	0.5920
30	0.2250	0.4227	0.3633	0.0570	0.4163	0.3603	0.1807	0.5877	0.5793

3.3 Evaluation methodology

This subsection describes in detail how we conducted the experiments for the different retrieval methods.

3.3.1 Approximation Errors of the Mapping Functions

In our first set of experiments we evaluated our hypothesis that the mapping between retrieval status values and probabilities relevance $Pr(\text{rel}|q, d)$ can be approximated by a logistic function. For this purpose, we performed an upper-bound experiment where we have complete relevance judgements, and then measured the goodness of fit of the two models.

We randomly chose 10 out of the 100 TREC-123 collections. The characteristics of these 10 collections are rather different but representative for the whole 100 collection test-bed. In addition, we restrict the experiments to TREC topics 51–100.

³<http://www.ucc.ie/gnuplot/gnuplot.html>

Figure 2: Baseline experiments: Recall/precision

(a) short queries

(b) mid-length queries

(c) long queries

For each query q (and each of the 10 collections as well as for each of the retrieval methods), we learned query-specific parameters $P(q) := \{b_{0,log}, b_{1,log}, c_{0,alin}, c_{1,alin}, c_{1,lin}\}$. Then, we computed the approximation error of the three mapping functions for q and the corresponding parameters $P(q)$. By this, we measure the approximation quality of the optimum parameters (the best parameters we can learn for this query); this allows us to check whether a mapping function is appropriate for approximating the actual relationship between RSV and probability of relevance. Later, we use a more realistic setting for evaluating retrieval quality in distributed IR.

For finding out the best mapping function, we computed the mean square approximation error of the relevance judgement for all of the 10 selected collections for short, mid-length and long queries (pure linear, affine linear and logistic mapping function).

As we are more interested in the top-ranked documents, we also plotted the mean square error when retrieving 1, ..., 100 documents. The average is computed over all 10 selected collections and topics 51–100; computation is done separately for short, mid-length and long queries.

We also conducted significance tests (t-Test) for the mean square error in the first 100 ranks. As common in t-Tests, we choose $p = 0.05$. With 100 error values (one for each rank) in each sample, the critical value is $t = 1.96$. Except for two collections used with TF-IDF, the differences in the mean square error are significant.

The reported differences between the pure/affine linear and logistic mapping functions are slightly biased in favour of the linear function, as we map results of the linear function outside $[0, 1]$ onto the corresponding margin (as we learned parameters $c_0, c_1 \in \mathbb{R}$ for the linear mapping function, the range is not restricted to $[0, 1]$). Without this mapping, the quality is slightly worse. As a consequence, the lower quality of the linear approximation of the relationship between the probability of inference and the probability of relevance cannot be blamed on the fact that the results of the linear function are not always in $[0, 1]$.

3.3.2 Retrieval Quality in Distributed IR

In our second set of experiments, we want to test whether a logistic relationship function improves quality for distributed IR.

All 100 TREC-123 collections are used for this experiment. For every collection, we learned one set of mapping function parameters with TREC topics 51–100 and evaluated this set on topics 101–150, and vice versa (cross evaluation). In this experiment, we are not interested in the influence of resource selection, thus retrieval is performed always on all collections (without resource selection), the top 30 documents of each collection are retrieved, and the resulting 3,000 documents are merged according to the approximations of $Pr(rel|q, d)$ (with collection- and retrieval-method-specific parameters).

We present both average precision at ranks 5, 10, 15, 20 and 30 as well as recall-precision graphs for the linear, the affine linear and the logistic mapping function.

We also conducted significance tests (t-Test) for the precision in the top ranks. Again, we choose $p = 0.05$. With 100 precision values (one for each query) in each sample, the critical value is $t = 1.96$. Differences which are significant are marked with an asterisk in the table.

3.4 Retrieval function: TF-IDF

In this subsection we report the results for the TF-IDF retrieval method.

3.4.1 Approximation Errors of the Mapping Functions

The mean square approximation error of the mapping functions is depicted in Tab. 3. The affine linear function performs better than the pure linear function. In addition, the logistic function outperforms the affine linear function in most cases (see Tab. 3(b)) the largest improvement is 12.8% for collection ap89_5 and long queries. In the same case, the logistic function performs 17.1% better than the pure linear function (see Tab. 3(c) for details). Except for patn3_5 with short queries and ziff1_2 for short and mid-length queries, the differences in the mean square errors of the two linear functions are significant. Furthermore, the errors of the linear functions and the logistic functions differ significantly.

Figure 3 shows the mean square error when retrieving $1, \dots, 100$ documents, averaged over all 10 selected collections and topics 51–100. One can see that the logistic function outperforms the two linear functions. Among the latter, the affine linear function is the better one. The difference in the approximation error is bigger for the top-ranked documents (which are typically the most interesting documents in retrieval); it decreases for the low-ranked documents. Here, most documents are irrelevant, and the mapping functions return a probability very close to zero, thus leading to a very low square error.

These results show that the approximation can be improved by using a logistic function.

3.4.2 Retrieval Quality in Distributed IR

Results for distributed IR with TF-IDF are depicted in Tab. 4.

Table 4 shows the average precision at ranks 5, 10, 15, 20 and 30 (averaged over all collections and all topics). The logistic function is compared with the affine linear one in Tab. 4(b) and with the pure linear one in Tab. 4(c). Here, the logistic function performs better than the pure linear function in most cases; the quality of the logistic mapping function is about the same as for the affine linear one (better in some cases and worse in other ones). In a few cases, the affine linear function is worse than the pure linear one. This is then due to the fact that we learn parameters for both functions independently on the training set and evaluate them on other queries (which might not be the optimal parameters for the evaluation query set). In general, the differences are not significant (t-Test with $p = 0.05$).

We do not present recall-precision graphs for the three mapping functions as they are very close together for all query lengths.

3.5 Retrieval function: INQUERY

In this subsection we present the INQUERY results.

3.5.1 Approximation Errors of the Mapping Functions

Mean square approximation errors of all mapping functions are listed in Tab. 5. Using a logistic function instead of a pure or affine linear one always reduces the mean square error; for the affine linear function, the improvement ranges between 4.2% (collection ziff3_9, mid-length queries) and 35.6% (collection ap89_5, long queries). The improvement is substantial; it is higher for long queries than for short and mid-length queries; the same is true for the absolute approximation error itself (probably due to the fact that the retrieval algorithms performs better for long queries, i.e. more relevant documents are top-ranked).

In addition, Fig. 4 shows the mean square error when retrieving $1, \dots, 100$ documents, averaged over all 10 selected collections and topics 51–100. One can see that the logistic function performs much better for the top-ranked documents; as for TF-IDF, the difference between the two approximation errors decreases for the low-ranked documents. These results show a clear improvement for the logistic function over the linear ones, especially for the most interesting ranks.

The approximation error of the linear functions is higher than for TF-IDF, in particular for the first documents. The error for the logistic function is about the same as for TF-IDF in the top ranks, and slightly higher than TF-IDF for the lower ranks.

3.5.2 Retrieval Quality in Distributed IR

Results for the case of distributed retrieval are depicted in Tab. 6 and Fig. 5.

Average precision at ranks 5, 10, 15, 20 and 30 is given in Tab. 6. The results show a significant improvement when an affine linear function is used instead of a pure linear one (particularly for mid-length queries). Precision in the top ranks can also be increased (but not significantly) by using a logistic mapping function instead of an affine linear one, although this improvement is quite small (and not significant), and in a few cases—especially for short queries—precision also decreases. The difference between the pure linear and the affine linear function is significant for mid-length and long queries.

In addition, recall-precision graphs are given in Fig. 5. Here the recall-precision graphs of the affine linear function are much better than the graphs for the pure linear one (especially for the mid-length and

Table 3: TF-IDF: mean square approximation error (in 10^{-3}) and improvement (in %)

(a) Pure linear function ./ affine linear function

Collection	size	short queries			mid-length queries			long queries		
		lin.	alin.	Δ	lin.	alin.	Δ	lin.	alin.	Δ
ap88_6	9210	310	302	+2.8	132	129	+2.4	074	071	+4.7
ap89_5	10542	351	339	+3.7	146	142	+2.5	082	078	+4.9
ap90_4	9955	604	589	+2.5	260	255	+2.2	152	145	+4.2
ap90_6	8394	476	464	+2.4	208	204	+1.9	125	121	+3.5
ap90_7	9794	545	529	+2.8	232	229	+1.2	134	129	+3.5
patn3_5	1021	167	165	+0.7	080	079	+1.1	054	054	+0.4
wsj88_3	13602	357	351	+1.8	135	134	+1.3	072	070	+2.5
wsj90_2	10560	372	366	+1.7	159	157	+1.1	094	092	+1.6
ziff1_2	8847	203	203	+0.2	082	082	+0.1	048	048	+0.1
ziff3_9	9670	230	228	+0.7	091	091	+0.1	055	055	+0.1

(b) Affine linear function ./ logistic function

Collection	size	short queries			mid-length queries			long queries		
		alin.	log.	Δ	alin.	log.	Δ	alin.	log.	Δ
ap88_6	9210	302	292	+3.3	129	125	+2.7	071	062	+12.0
ap89_5	10542	339	326	+3.6	142	134	+5.8	078	068	+12.8
ap90_4	9955	589	588	+0.3	255	248	+2.5	145	134	+7.6
ap90_6	8394	464	463	+0.3	204	200	+1.7	121	113	+6.2
ap90_7	9794	529	524	+1.0	229	228	+0.8	129	121	+5.9
patn3_5	1021	165	164	+0.8	079	078	+1.0	054	052	+2.6
wsj88_3	13602	351	355	-1.2	134	134	+0.0	070	068	+2.8
wsj90_2	10560	366	364	+0.6	157	158	-0.7	092	090	+2.7
ziff1_2	8847	203	210	-3.4	082	083	-1.7	048	050	-2.8
ziff3_9	9670	228	232	-1.8	091	092	-1.7	055	057	-2.0

(c) Pure linear function ./ logistic function

Collection	size	short queries			mid-length queries			long queries		
		lin.	log.	Δ	lin.	log.	Δ	alin.	log.	Δ
ap88_6	9210	310	292	+6.1	132	125	+5.0	074	062	+16.1
ap89_5	10542	351	326	+7.1	146	134	+8.2	082	068	+17.1
ap90_4	9955	604	588	+2.7	260	248	+4.7	152	134	+11.5
ap90_6	8394	476	463	+2.7	208	200	+3.6	125	113	+9.5
ap90_7	9794	545	524	+3.8	232	228	+2.0	134	121	+9.2
patn3_5	1021	167	164	+1.5	080	078	+2.1	054	052	+3.0
wsj88_3	13602	357	355	+0.7	135	134	+1.3	072	068	+5.2
wsj90_2	10560	372	364	+2.3	159	158	+0.4	094	090	+4.2
ziff1_2	8847	203	210	-3.2	082	083	-1.6	048	050	-2.7
ziff3_9	9670	230	232	-1.0	091	092	-1.5	055	057	-2.0

Figure 3: TF-IDF: Mean square approximation error when retrieving 1, ..., 100 documents

(a) short queries

(b) mid-length queries

(c) long queries

Table 4: TF-IDF: Average precision at given ranks, improvement (in %) and t-test (p=0.05)

(a) Pure linear function ./ affine linear function

Rank	short queries			mid-length queries			long queries		
	lin.	alin.	Δ	lin.	alin.	Δ	lin.	alin.	Δ
5	0.3100	0.3320	+7.1	0.3220	0.3300	+2.5	0.4660	0.4600	-1.3
10	0.2970	0.3050	+2.7	0.3040	0.2980	-2.0	0.4390	0.4390	+0.0
15	0.2887	0.2933	+1.6	0.2927	0.3007	+2.7	0.4113	0.4213	+2.4
20	0.2860	0.2900	+1.4	0.2815	0.2880	+2.3	0.4085	0.4120	+0.9
30	0.2803	0.2807	+0.1	0.2703	0.2717	+0.5	0.3967	0.3983	+0.4

(b) Affine linear function ./ logistic function

Rank	short queries			mid-length queries			long queries		
	alin.	log.	Δ	alin.	log.	Δ	alin.	log.	Δ
5	0.3320	0.3420	+3.0	0.3300	0.3280	-0.6	0.4600	0.4400	-4.3
10	0.3050	0.3210	+5.2	0.2980	0.2980	+0.0	0.4390	0.4350	-0.9
15	0.2933	0.3047	+3.9	0.3007	0.2967	-1.3	0.4213	0.4273	+1.4
20	0.2900	0.2885	-0.5	0.2880	0.2870	-0.3	0.4120	0.4185	+1.6
30	0.2807	0.2773	-1.2	0.2717	0.2737	+0.7	0.3983	0.4040	+1.4

(c) Pure linear function ./ logistic function

Rank	short queries				mid-length queries			long queries		
	lin.	log.	Δ		lin.	log.	Δ	lin.	log.	Δ
5	0.3100	0.3420	+10.3	*	0.3220	0.3280	+1.9	0.4660	0.4400	-5.6
10	0.2970	0.3210	+8.1	*	0.3040	0.2980	-2.0	0.4390	0.4350	-0.9
15	0.2887	0.3047	+5.5		0.2927	0.2967	+1.4	0.4113	0.4273	+3.9
20	0.2860	0.2885	+0.9		0.2815	0.2870	+2.0	0.4085	0.4185	+2.4
30	0.2803	0.2773	-1.1		0.2703	0.2737	+1.3	0.3967	0.4040	+1.8

Table 5: INQUERY: mean square approximation error (in 10^{-3}) and improvement (in %)

(a) Pure linear function ./ affine linear function

Collection	size	short queries			mid-length queries			long queries		
		lin.	alin.	Δ	lin.	alin.	Δ	lin.	alin.	Δ
ap88_6	9210	333	309	+7.2	141	136	+3.4	083	079	+5.6
ap89_5	10542	383	339	+11.4	159	153	+3.7	093	086	+7.0
ap90_4	9955	635	581	+8.6	276	260	+5.8	165	154	+6.6
ap90_6	8394	498	459	+7.9	215	207	+3.9	132	126	+4.6
ap90_7	9794	555	525	+5.5	239	233	+2.5	142	136	+4.6
patn3_5	1021	164	158	+4.1	078	076	+1.8	055	053	+3.6
wsj88_3	13602	375	351	+6.3	142	137	+3.4	077	074	+4.7
wsj90_2	10560	385	367	+4.6	164	159	+2.9	097	094	+3.0
ziff1_2	8847	212	206	+2.7	083	082	+1.5	049	048	+2.2
ziff3_9	9670	241	232	+3.8	090	088	+2.3	056	054	+3.3

(b) Affine linear function ./ logistic function

Collection	size	short queries			mid-length queries			long queries		
		alin.	log.	Δ	alin.	log.	Δ	alin.	log.	Δ
ap88_6	9210	309	272	+12.1	136	121	+11.1	079	058	+26.3
ap89_5	10542	339	290	+14.6	153	127	+17.1	086	055	+35.6
ap90_4	9955	581	519	+10.7	260	228	+12.2	154	121	+21.7
ap90_6	8394	459	415	+9.4	207	187	+9.6	126	098	+22.1
ap90_7	9794	525	488	+7.0	233	216	+7.4	136	107	+21.0
patn3_5	1021	158	129	+18.1	076	071	+7.0	053	049	+8.9
wsj88_3	13602	351	314	+10.5	137	118	+13.6	074	054	+27.1
wsj90_2	10560	367	340	+7.4	159	145	+9.2	094	079	+16.4
ziff1_2	8847	206	188	+8.7	082	076	+6.5	048	042	+13.6
ziff3_9	9670	232	222	+4.1	088	084	+4.2	054	051	+6.0

(c) Pure linear function ./ logistic function

Collection	size	short queries			mid-length queries			long queries		
		lin.	log.	Δ	lin.	log.	Δ	alin.	log.	Δ
ap88_6	9210	333	272	+18.4	141	121	+14.1	083	058	+30.4
ap89_5	10542	383	290	+24.4	159	127	+20.2	093	055	+40.1
ap90_4	9955	635	519	+18.3	276	228	+17.3	165	121	+26.9
ap90_6	8394	498	415	+16.6	215	187	+13.1	132	098	+25.7
ap90_7	9794	555	488	+12.1	239	216	+9.8	142	107	+24.6
patn3_5	1021	164	129	+21.5	078	071	+8.7	055	049	+12.2
wsj88_3	13602	375	314	+16.2	142	118	+16.6	077	054	+30.6
wsj90_2	10560	385	340	+11.6	164	145	+11.9	097	079	+18.9
ziff1_2	8847	212	188	+11.1	083	076	+7.9	049	042	+15.5
ziff3_9	9670	241	222	+7.8	090	084	+6.4	056	051	+9.1

Figure 4: INQUERY: Mean square approximation error when retrieving 1, ..., 100 documents

(a) short queries

(b) mid-length queries

(c) long queries

Table 6: INQUERY: Average precision at given ranks, improvement (in %) and t-test (p=0.05)

(a) Pure linear function ./ affine linear function

Rank	short queries				mid-length queries				long queries			
	lin.	alin.	Δ		lin.	alin.	Δ		lin.	alin.	Δ	
5	0.3960	0.4300	+8.6	*	0.3680	0.4360	+18.5	*	0.5380	0.5660	+5.2	
10	0.3910	0.4270	+9.2	*	0.3430	0.4070	+18.7	*	0.5120	0.5480	+7.0	*
15	0.3827	0.4140	+8.2	*	0.3293	0.3893	+18.2	*	0.4907	0.5313	+8.3	*
20	0.3660	0.4060	+10.9	*	0.3250	0.3770	+16.0	*	0.4820	0.5250	+8.9	*
30	0.3473	0.3880	+11.7	*	0.3050	0.3653	+19.8	*	0.4637	0.5113	+10.3	*

(b) Affine linear function ./ logistic function

Rank	short queries				mid-length queries				long queries			
	lin.	log.	Δ		alin.	log.	Δ		alin.	log.	Δ	
5	0.4300	0.4340	+0.9		0.4360	0.4480	+2.8		0.5660	0.5580	-1.4	
10	0.4270	0.3980	-6.8		0.4070	0.4190	+2.9		0.5480	0.5580	+1.8	
15	0.4140	0.3913	-5.5		0.3893	0.4087	+5.0		0.5313	0.5380	+1.3	
20	0.4060	0.3860	-4.9		0.3770	0.3955	+4.9		0.5250	0.5260	+0.2	
30	0.3880	0.3823	-1.5		0.3653	0.3813	+4.4		0.5113	0.5153	+0.8	

(c) Pure linear function ./ logistic function

Rank	short queries				mid-length queries				long queries			
	lin.	log.	Δ		lin.	log.	Δ		lin.	log.	Δ	
5	0.3960	0.4340	+9.6	*	0.3680	0.4480	+21.7	*	0.5380	0.5580	+3.7	
10	0.3910	0.3980	+1.8		0.3430	0.4190	+22.2	*	0.5120	0.5580	+9.0	*
15	0.3827	0.3913	+2.2		0.3293	0.4087	+24.1	*	0.4907	0.5380	+9.6	*
20	0.3660	0.3860	+5.5		0.3250	0.3955	+21.7	*	0.4820	0.5260	+9.1	*
30	0.3473	0.3823	+10.1	*	0.3050	0.3813	+25.0	*	0.4637	0.5153	+11.1	*

Figure 5: INQUERY: Recall/precision for logistic and linear function

(a) short queries

(b) mid-length queries

(c) long queries

long queries). The difference between the graphs for the affine linear function and the logistic function is rather small, however on average we obtained slight improvements by using the logistic function.

Both—the table and the figure—show that distributed IR with INQUERY yields a better performance than using TF-IDF, repeating the results from section 3.2 that INQUERY is a better retrieval engine than TF-IDF.

3.6 Retrieval function: LM

This subsection contains the results for language models (LM).

3.6.1 Approximation Errors of the Mapping Functions

Table 7 shows the mean square approximation errors of all mapping functions for LM (language models). Switching from a pure linear function to an affine linear function (i.e. adding a second degree of freedom) clearly improves the approximation quality. The maximum improvement is 19.3% for collection ap89_5, short queries. We can dramatically reduce the approximation error by using a logistic function instead of an affine linear one; here the improvement is maximised for ap89_5 for long queries (34.5%). These results are also supported by Fig. 6 (mean square error when retrieving 1, ..., 100 documents): The logistic mapping function yields a much higher accuracy than the linear functions. The difference is maximised for the top-ranked documents (the most important ones). For lower ranks, the approximation quality increases, as here most documents are not relevant, and the functions map to a probability close to zero.

The improvement and the approximation errors are slightly worse than for INQUERY, especially for the pure linear function. The differences between this pure linear function and the two other functions are higher for LM than for INQUERY.

3.6.2 Retrieval Quality in Distributed IR

Table 8 contains average precision at ranks 5, 10, 15, 20 and 30. The logistic function is compared with the affine linear one in Tab. 8(b) and with the pure linear one in Tab. 8(c). Interestingly, the pure linear function performs extraordinarily bad and—thus—does not seem to be suitable as a mapping function for language models. The logistic function performs better than the affine linear function, in particular for short queries (for which the difference is significant for $p = 0.05$).

These results are confirmed by the recall-precision graphs in Fig. 7. For all three query types, the logistic mapping function outperforms the affine linear one. The retrieval quality of the pure linear function is poor.

4 Conclusion

In this paper, we investigated the relationship between the RSV of different retrieval engines and the probability of relevance $Pr(\text{rel}|q, d)$. In the past, little effort has been spent on estimating the latter, since the most popular retrieval task—ad-hoc retrieval—needs only a monotonic function of this probability in order to yield a ranking according to the probability ranking principle.

However, computing probabilities of relevance as a normalisation of the RSVs allows for comparing the results from different retrieval engines or indexing weights. Actually, advanced IR applications are based on estimates of the actual probabilities of relevance, e.g. for approximating the number of relevant documents in the result set for resource selection (decision-theoretic framework [14]) or for merging the documents retrieved from the selected collections into a single ranked list.

For the specific case of uncertain inference, Rijsbergen proposed a linear mapping function for modelling the relationship between the probability of inference and the probability of relevance. In [15] we showed that this approach has only a moderate approximation quality, in particular in the top-ranked documents.

In this paper we generalised our evaluation. We considered different retrieval engines: TF-IDF, INQUERY and language models (LM). Besides linear mapping functions, we proposed the use of a logistic function. The logistic function can be justified from a theoretical point of view, as it is a continuous approximation of the discrete step function (the ideal relationship).

Table 7: LM: mean square approximation error (in 10^{-3}) and improvement (in %)

(a) Pure linear function ./ affine linear function

Collection	size	short queries			mid-length queries			long queries		
		lin.	alin.	Δ	lin.	alin.	Δ	lin.	alin.	Δ
ap88_6	9210	364	314	+13.7	146	138	+5.1	089	080	+9.9
ap89_5	10542	432	349	+19.3	167	156	+6.2	101	088	+12.5
ap90_4	9955	715	581	+18.7	296	264	+10.9	181	158	+12.8
ap90_6	8394	536	464	+13.6	223	209	+6.0	139	128	+7.9
ap90_7	9794	595	529	+11.1	245	235	+4.1	150	138	+8.0
patn3_5	1021	176	161	+8.6	081	076	+6.7	060	053	+10.3
wsj88_3	13602	405	355	+12.3	148	139	+5.8	083	075	+9.6
wsj90_2	10560	405	370	+8.8	169	161	+4.5	101	095	+5.6
ziff1_2	8847	222	210	+5.3	085	082	+2.9	051	049	+4.7
ziff3_9	9670	256	231	+9.8	093	087	+5.8	059	053	+9.2

(b) Affine linear function ./ logistic function

Collection	size	short queries			mid-length queries			long queries		
		alin.	log.	Δ	alin.	log.	Δ	alin.	log.	Δ
ap88_6	9210	314	281	+10.5	138	126	+8.8	080	060	+24.9
ap89_5	10542	349	302	+13.4	156	133	+15.0	088	058	+34.5
ap90_4	9955	581	525	+9.6	264	231	+12.4	158	120	+23.9
ap90_6	8394	464	427	+8.0	209	190	+9.3	128	099	+22.7
ap90_7	9794	529	495	+6.4	235	221	+6.1	138	108	+21.5
patn3_5	1021	161	145	+9.7	076	070	+8.2	053	049	+8.0
wsj88_3	13602	355	312	+12.3	139	121	+13.1	075	054	+28.0
wsj90_2	10560	370	339	+8.2	161	145	+10.2	095	078	+17.9
ziff1_2	8847	210	199	+5.1	082	079	+3.6	049	043	+11.4
ziff3_9	9670	231	220	+4.7	087	083	+4.6	053	048	+9.3

(c) Pure linear function ./ logistic function

Collection	size	short queries			mid-length queries			long queries		
		lin.	log.	Δ	lin.	log.	Δ	alin.	log.	Δ
ap88_6	9210	364	281	+22.8	146	126	+13.5	089	060	+32.3
ap89_5	10542	432	302	+30.2	167	133	+20.3	101	058	+42.7
ap90_4	9955	715	525	+26.5	296	231	+22.0	181	120	+33.7
ap90_6	8394	536	427	+20.5	223	190	+14.8	139	099	+28.8
ap90_7	9794	595	495	+16.8	245	221	+9.9	150	108	+27.8
patn3_5	1021	176	145	+17.5	081	070	+14.4	060	049	+17.5
wsj88_3	13602	405	312	+23.1	148	121	+18.1	083	054	+34.9
wsj90_2	10560	405	339	+16.3	169	145	+14.3	101	078	+22.5
ziff1_2	8847	222	199	+10.1	085	079	+6.4	051	043	+15.6
ziff3_9	9670	256	220	+14.1	093	083	+10.2	059	048	+17.7

Figure 6: LM: Mean square approximation error when retrieving 1, ..., 100 documents

(a) short queries

(b) mid-length queries

(c) long queries

Table 8: LM: Average precision at given ranks, improvement (in %) and t-test (p=0.05)

(a) Pure linear function Δ . affine linear function												
Rank	short queries			Δ	mid-length queries			Δ	long queries			Δ
	lin.	alin.	lin.		alin.	lin.	alin.					
5	0.0740	0.3580	+383.8	*	0.0580	0.3140	+441.4	*	0.0840	0.5620	+569.0	*
10	0.0700	0.3620	+417.1	*	0.0610	0.3190	+423.0	*	0.0840	0.5300	+531.0	*
15	0.0713	0.3527	+394.7	*	0.0687	0.3193	+364.8	*	0.1060	0.5020	+373.6	*
20	0.0755	0.3370	+346.4	*	0.0715	0.3115	+335.7	*	0.1145	0.4865	+324.9	*
30	0.0817	0.3310	+305.1	*	0.0817	0.2827	+246.0	*	0.1363	0.4653	+241.4	*

(b) Affine linear function Δ . logistic function												
Rank	short queries			Δ	mid-length queries			Δ	long queries			Δ
	alin.	log.	alin.		log.	alin.	log.					
5	0.3580	0.4200	+17.3	*	0.3140	0.3380	+7.6		0.5620	0.5680	+1.1	
10	0.3620	0.3970	+9.7	*	0.3190	0.3200	+0.3		0.5300	0.5490	+3.6	
15	0.3527	0.3820	+8.3	*	0.3193	0.3140	-1.7		0.5020	0.5320	+6.0	*
20	0.3370	0.3700	+9.8	*	0.3115	0.3110	-0.2		0.4865	0.5205	+7.0	*
30	0.3310	0.3563	+7.6	*	0.2827	0.2993	+5.9		0.4653	0.5143	+10.5	*

(c) Pure linear function Δ . logistic function												
Rank	short queries			Δ	mid-length queries			Δ	long queries			Δ
	lin.	log.	lin.		log.	lin.	log.					
5	0.0740	0.4200	+467.6	*	0.0580	0.3380	+482.8	*	0.0840	0.5680	+576.2	*
10	0.0700	0.3970	+467.1	*	0.0610	0.3200	+424.6	*	0.0840	0.5490	+553.6	*
15	0.0713	0.3820	+435.8	*	0.0687	0.3140	+357.1	*	0.1060	0.5320	+401.9	*
20	0.0755	0.3700	+390.1	*	0.0715	0.3110	+335.0	*	0.1145	0.5205	+354.6	*
30	0.0817	0.3563	+336.1	*	0.0817	0.2993	+266.3	*	0.1363	0.5143	+277.3	*

Figure 7: LM: Recall/precision for logistic and linear function

(a) short queries

(b) mid-length queries

(c) long queries

Our experiments showed that for all retrieval methods, the pure linear function performs worst, and the logistic function performs best. The mean square errors of the three mapping functions differ significantly (for a specific retrieval method and query length). The improvements in the approximation error differ notably across the retrieval methods (given a fixed mapping function and query type). For INQUERY and LM, the improvement by using a logistic function is bigger than for TF-IDF. In general, long queries result in a better approximation quality.

In the application area of distributed IR without resource selection, using a logistic function yields slightly (not but significantly) higher precision in the top ranks than using linear functions; the pure linear function performs significantly worst (for LM, extraordinarily bad). The recall-precision graphs of TF-IDF are close together for all three mapping functions. For INQUERY, the affine linear function and the logistic function perform about the same; for LM, effectiveness is significantly higher when using a logistic function. As in the ad-hoc retrieval task, INQUERY performs also better than TF-IDF and LM in distributed IR.

Concluding, using a logistic mapping function instead of a linear one leads to a significant improvement for the approximation quality and a small improvement for the distributed retrieval quality.

In future, we will investigate the learning step of the mapping function parameters in more detail. In this paper, we learned parameters with 50 queries. The interesting question is how many documents per query and how many queries are really required for obtaining good parameters, and how the quality of the parameters is related to the size of the learning data.

Furthermore, we will search for better mapping functions. In our experiments, the logistic mapping function performs best, but there is still room for improvement.

5 Acknowledgements

This work is supported by the EU commission under grant IST-2000-26061 (project MIND).

Part of this work was performed while both authors were affiliated at the University of Dortmund.

We wish to thank Paul Ogilvie and Luo Si from CMU for their assistance with the Lemur toolkit.

References

- [1] J. Allan, J. Callan, W. Croft, L. Ballesteros, D. Byrd, R. Swan, and J. Xu. Inquery does battle with TREC-6. In *Proceedings of the Sixth Text REtrieval Conference (TREC-6)*, pages 169–206, Gaithersburg, MD, 1998. National Institute of Standards and Technology (NIST).
- [2] J. Callan and M. Connell. Query-based sampling of text databases. *ACM Transactions on Information Systems*, 19(2):97–130, 2001.
- [3] J. Callan, W. Croft, and S. Harding. The INQUERY retrieval system. In *Proceedings of DEXA-92, 3rd International Conference on Database and Expert Systems Applications*, pages 78–83, Heidelberg et al., 1992. Springer.
- [4] W. Cooper, F. Gey, and D. Dabney. Probabilistic retrieval based on staged logistic regression. In N. J. Belkin, P. Ingwersen, and A. M. Pejtersen, editors, *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Copenhagen, Denmark, June 21-24, 1992*, pages 198–210, New York, 1992. ACM.
- [5] S. Fienberg. *The Analysis of Cross-Classified Categorical Data*. MIT Press, Cambridge, Mass., 2. edition, 1980.
- [6] D. Freeman. *Applied Categorical Data Analysis*. Dekker, New York, 1987.
- [7] N. Fuhr and C. Buckley. A probabilistic learning approach for document indexing. *ACM Transactions on Information Systems*, 9(3):223–248, 1991.

- [8] N. Fuhr and U. Pfeifer. Combining model-oriented and description-oriented approaches for probabilistic indexing. In *Proceedings of the Fourteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 46–56, New York, 1991. ACM.
- [9] F. C. Gey. Inferring probability of relevance using the method of logistic regression. In B. W. Croft and C. J. van Rijsbergen, editors, *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 222–231, London, et al., 1994. Springer-Verlag.
- [10] L. Gravano and H. Garcia-Molina. Generalizing GIOSS to vector-space databases and broker hierarchies. In U. Dayal, P. Gray, and S. Nishio, editors, *VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases*, pages 78–89, Los Altos, California, 1995. Morgan Kaufman.
- [11] L. Gravano, H. Garcia-Molina, and A. Tomasic. The effectiveness of GIOSS for the text database discovery problem. In R. T. Snodgrass and M. Winslett, editors, *Proceedings of the 1994 ACM SIGMOD. International Conference on Management of Data.*, pages 126–137, New York, 1994. ACM.
- [12] D. Harman, editor. *The Second Text REtrieval Conference (TREC-2)*, Gaithersburg, Md. 20899, 1994. National Institute of Standards and Technology.
- [13] R. Manmatha, T. Rath, and F. Feng. Modeling score distributions for combining the outputs of search engines. In W. Croft, D. Harper, D. Kraft, and J. Zobel, editors, *Proceedings of the 24th Annual International Conference on Research and development in Information Retrieval*, New York, 2001. ACM.
- [14] H. Nottelmann and N. Fuhr. Evaluating different methods of estimating retrieval quality for resource selection. In *Proceedings of the 26st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, 2003. ACM.
- [15] H. Nottelmann and N. Fuhr. From uncertain inference to probability of relevance for advanced IR applications. In F. Sebastiani, editor, *25th European Conference on Information Retrieval Research (ECIR 2003)*, pages 235–250, Heidelberg et al., 2003. Springer.
- [16] M. Pollmann. Entwicklung und Untersuchung von verbesserten probabilistischen Indexierungsfunktionen für Freitext-Indexierung (in german). Master's thesis, Universität Dortmund, Fachbereich Informatik, 1993.
- [17] J. Ponte and W. Croft. A language modeling approach to information retrieval. In W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson, and J. Zobel, editors, *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281, New York, 1998. ACM.
- [18] M. Porter. An algorithm for suffix stripping. *Program*, 14:130–137, July 1980.
- [19] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, editors. *Nested Relations and Complex Objects in Databases*. Cambridge University Press, 1992.
- [20] S. Robertson. The probability ranking principle in IR. *Journal of Documentation*, 33:294–304, 1977.
- [21] G. Salton, editor. *The SMART Retrieval System - Experiments in Automatic Document Processing*. Prentice Hall, Englewood, Cliffs, New Jersey, 1971.
- [22] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.
- [23] C. J. van Rijsbergen. A non-classical logic for information retrieval. *The Computer Journal*, 29(6):481–485, 1986.
- [24] C. J. van Rijsbergen. Probabilistic retrieval revisited. *The Computer Journal*, 35(3):291–298, 1992.

- [25] S. Wong and Y. Yao. On modeling information retrieval with probabilistic inference. *ACM Transactions on Information Systems*, 13(1):38–68, 1995.