A Signal-to-Noise Approach to Score Normalization

Avi Arampatzis Jaap Kamps
Archives and Information Studies, Media Studies
University of Amsterdam, The Netherlands
avi.arampatzis@gmail.com kamps@uva.nl

ABSTRACT

Score normalization is indispensable in distributed retrieval and fusion or meta-search where merging of result-lists is required. Distributional approaches to score normalization with reference to relevance, such as binary mixture models like the normal-exponential, suffer from lack of universality and troublesome parameter estimation especially under sparse relevance. We develop a new approach which tackles both problems by using aggregate score distributions without reference to relevance, and is suitable for uncooperative engines. The method is based on the assumption that scores produced by engines consist of a signal and a noise component which can both be approximated by submitting well-defined sets of artificial queries to each engine. We evaluate in a standard distributed retrieval testbed and show that the signal-to-noise approach yields better results than other distributional methods. As a significant by-product, we investigate query-length distributions.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.2.4 [**Database Management**]: Systems—*query processing*

General Terms

Experimentation, Performance, Theory

Keywords

Distributed retrieval, resource selection, meta-search, fusion, score normalization, score distribution, filtering, query length distribution, query model, Zipf's law, power-law.

1. INTRODUCTION

Modern best-match retrieval models calculate some kind of score per collection item which serves as a measure of the degree of relevance to an input request. Scores are used in ranking retrieved items. Their range and distribution varies wildly across different models making them incomparable across different engines [25], even across different requests on the same engine if they are influenced by non-semantic query characteristics, e.g. length. Even most probabilistic models do not calculate the probability of relevance of items directly, but some order-preserving (monotone or isotone) function of it [21].

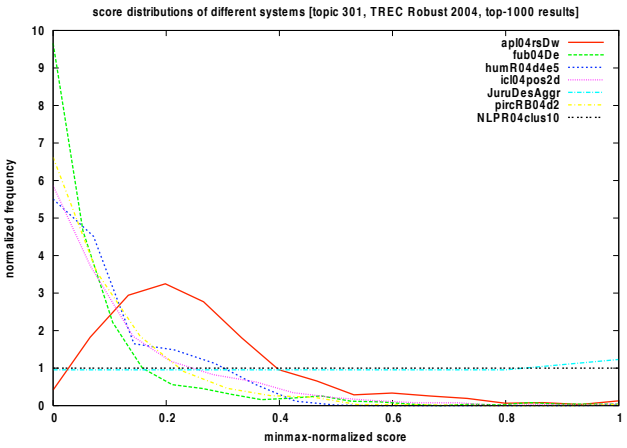

Figure 1: Min-max normalized score distributions (top-1,000 results) of different systems for TREC topic 301 in Robust Track 2004. Score outputs would be even more diverse for disjoint collections, especially across different data-types beyond text.

For single-collection ad-hoc retrieval, the variety of score types is not an issue; scores do not have to be comparable across models and requests, since they are only used to rank items per request per system. However, in advanced applications, such as distributed retrieval, fusion, or applications requiring thresholding such as filtering or recall-oriented search, some form of score normalization is imperative. In the first two applications, several rankings (with non-overlapping and overlapping sets of items respectively) have to be merged or fused to a single ranking. Here, score normalization is an important step [9]. In practice, while many users never use meta-search engines directly, most conventional search engines are faced with the problem of combining results from many sub-engines. For example, blending images, text, inline answers, stock quotes, and so on, has become common.

In filtering, bare scores give no indication on whether to retrieve an incoming document or not. Usually, a user model is captured into some evaluation measure. Some of these measures can be optimized by thresholding the probability of relevance at some specific level [19], thus a method of normalizing scores into probabilities is needed. Moreover, thresholding has turned out to be important in recall-oriented retrieval setups, such as legal or patent search,

CIKM 2009, Hong Kong, China, November 2–6, 2009.

where ranked retrieval has a particular disadvantage in comparison with traditional Boolean retrieval: there is no clear cut-off point where to stop consulting results [22]. Again, normalizing scores to expected values of a given effectiveness measure allows for optimal rank thresholding [5].

Popular methods, e.g. range normalization based on minimum and maximum scores, are rather naive, considering the wild variety of score outputs across search engines, because they do not take into account the *shape* of score distributions (SDs). Figure 1 demonstrates the variety of outputs across different systems for constant query and collection. Although these methods have worked reasonably well for merging or fusing results [18], more advanced approaches have been seen which try to improve normalization by investigating SDs. Such methods have been found to work at least as well (or in some cases better) than the simple ones in the context of fusion [20, 13]. They have also been found effective for thresholding in filtering [7, 29, 11] or thresholding ranked lists [5]. We are not aware of any empirical evidence in the context of distributed retrieval.

The main aim of this paper is to analyse and further develop score distributional approaches to score normalization. Our underlying assumption is that normalization methods that take the shape of the SD into account will be more effective than methods that ignore it. We want to make no assumptions on the search engines generating the scores to be normalized other than that they produce ranked lists sorted by decreasing score. Thus, we treat each engine as a 'black-box' and are interested in approaches based only on observing their input-output characteristics: the queries and resulting score distributions.

In Section 2 we examine the state of affairs in using mixture-model normalization methods to map scores to probabilities of relevance. We note theoretical as well as practical problems and limitations in the applicability of such methods. In Section 3 we investigate the single SD model of [12], arguing that some of its complexity is unnecessary. In Section 4 we take a novel approach and introduce three new methods. Two single SDs are used, representing score signal and noise, and input scores are normalized according to their signal to noise ratios. The two SDs are generated as score aggregates of two different types of artificial queries, human-like and noise respectively. For query generation, we develop two query models in Section 5. As a significant by-product, we investigate the length distributions of such input queries; theoretically for noise, and using real data for humans. In Section 6 we evaluate the methods in a standard distributed retrieval testbed. This is also the first evaluation of the currently most-popular mixture model in a distributed retrieval setup. Conclusions are drawn in Section 7.

2. MIXTURE MODELS

Under the assumption of binary relevance, classic attempts model SDs, on a per-request basis, as a mixture of two distributions: one for relevant and the other for non-relevant documents. Given the two component distributions and their mix weight, the probability of relevance of a document given its score can be calculated straightforwardly [7, 20], essentially allowing to normalize scores into probabilities of relevance. Furthermore, the expected numbers of relevant and non-relevant documents above and below any rank or score can be estimated, allowing to calculate precision, recall, or any other traditional measure at any given threshold enabling its optimization [5]. Assuming proper component choices, such methods are theoretically 'clean' and non-parametric.

Various combinations of distributions have been proposed since the early years of IR; for a recent extended review and theoretical analysis of the choices, we refer the reader to [25]. The currently most popular model being that of using a normal for relevant and an exponential for non-relevant, introduced in [3] and [7, 20] and followed up by [29, 11] and others. The latest improvements of the normal-exponential model use truncated versions of the component densities, trying to deal with some of its shortcomings [5]. In this study, we do not set out to investigate alternative mixtures, but just use the standard normal-exponential.

2.1 The Normal-Exponential Model

In this section, we investigate the theoretical as well as the empirical evidence for using a mixture of normal-exponential for modeling SDs in IR. We note theoretical as well as practical problems.

2.1.1 Normal for Relevant

A theorem claims that the distribution of relevant document scores converges to a Gaussian central limit (GCL) quickly, with "corrections" diminishing as O(1/k) where k is the query length [7]. Roughly, three assumptions were made: *a)* terms occurring independently, *b)* scores are calculated via some linear combination of document term weights, and *c)* relevant documents cluster around some point in the document space with some hyper-ellipsoidal density with fast-falling tails. Based on those assumptions, the proof is more likely to hold in setups with vector space or geometric models with dot-product or cosine similarity scoring and long queries. This does not mean that the Gaussian does not get along well with other retrieval models or setups, but we have not found any supportive theory in the literature.

Although the GCL is approached theoretically quickly as query length increases, practically, queries of length above a dozen terms are only possible through relevance feedback and other learning methods. For short queries, the Gaussian may simply not be there to be estimated. Empirically, using a vector space model with scores which were unbounded above on TREC data, [7] found usable Gaussian shapes to form at around $k = 250$. k also seemed to depend on the quality of a query; the better the query, the fewer the terms necessary for a normal approximation of the observed distribution. Along similar lines, [20] noticed that better systems (in terms of average precision) produce better Gaussian shapes.

2.1.2 Exponential for Non-relevant

Under a similar set of assumptions and approximations to the ones mentioned above, [7] investigate also the distribution of non-relevant document scores and conclude that a GCL is unlikely and if it appears it does only at a very slow rate with k—practically never seen even for massive query expansion. Nevertheless, such a theorem does not help much in determining a usable distribution. In absence of a related theory or a simpler method, the use of the exponential distribution has been so far justified empirically: it generally fits well to the *high-end* of non-relevant item scores, but not to all.

Different cutoffs have been used for fitting purposes: [7, 11] fit on the top 50–100, [20] fit on almost the top-1,000 (1,000 minus the number of relevant documents). [2] fits even on a non-uniform sample of the whole score range, but the approach is rather system/task-specific. In general, it is difficult to get a good exponential fit on the whole score range. Figure 2 shows the total score densities produced by a combination of two queries and two sub-collections of TREC-4 using KL-DIVERGENCE as a retrieval model. Obviously, none of these SDs can be fitted *in totality* with the mixture. Candidate ranges are usually $[s_{\text{peak}}, +\infty)$ where s_{peak} is set at the mode (or higher) of the total SD.

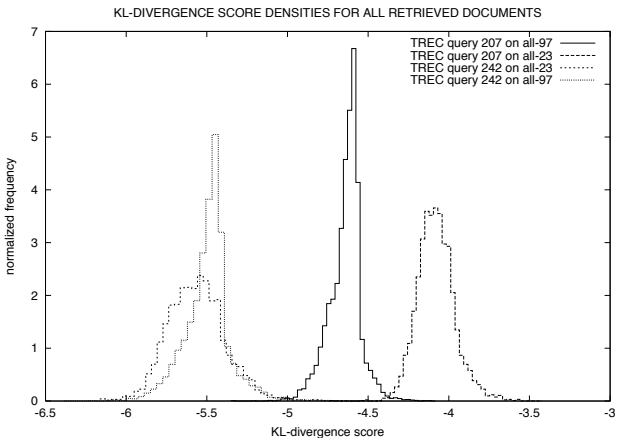

Figure 2: KL-divergence score densities; 2 queries on 2 collections.

2.1.3 Non-convexity

The normal-exponential model has a serious theoretical problem: the estimated probability of relevance as a function of the score is non-monotonic. This behavior is a direct result of the Gaussian falling more rapidly than the exponential and hence the two density functions intersect twice. In most cases, this non-monotonicity appears inside the effective score range making the probability of relevance decline as the score increases above some point. Robertson [25] formulates the problem as non-convexity of the recall-fallout curve as seen from recall=1, fallout=0.

In adaptive filtering, [3, 7] deal with the problem by selecting as threshold the lower solution of the 2nd degree equation resulting from optimizing linear utility measures, while [29, 11] do not seem to notice or deal with it. In meta-search, [20] noted the problem and *forced* the probability to be monotonic by drawing a straight line from the point where the probability is maximum to the point $[1, 1]$. Both procedures, although they may have been suitable for the above tasks, are theoretically unjustified.

In [5], the two component SDs were set to uniform within the offending score range; this is equivalent to randomization of the corresponding sub-ranking and it is justified as enforcing recall-fallout convexity. Nevertheless, this implies that the normal-exponential mixture is theoretically valid and that scoring formulas produce suboptimal rankings which can be improved by randomizing; evidence of such scoring schemes within modern text retrieval systems has not been found [6]. Consequently, we choose not to use any fixes in the experiments in this paper; we just assign the maximum-reached output probability to input scores affected by the non-monotonicity.[1]

2.1.4 Normal-Exponential in Practice

Despite the above-mentioned theoretical problems, the model was applied successfully in fusion, with short queries and even with a scoring system which produces scores between 0 and 1 without worrying about the implied truncation at both ends for the normal and at the high end for the exponential [20]. In the context of thresholding for document filtering, with the generally unbounded scoring function BM25 and a maximum of 60 query terms per profile, the method performed well (2nd best, after Maximum Likelihood Estimation) on 3 out of 4 TREC data sets [11]. We are not aware of any work related to the use of normal-exponential in distributed setups.

A recent study on the applicability of the mixture on a wide variety of modern text retrieval systems shows support for vector space or geometric models, as well as BM25, as being amenable to the normal-exponential. While longer queries tend to lead to smoother SDs and improved fits, the end-result in thresholding is better for the short title queries with high quality keywords [6].

2.2 Estimation and other Problems

All mixture models, irrespective of component choices, present some practical problems. Estimating the component densities is best done when many relevance judgements are available. In practice, relevance judgements are not available at all, or they are sparse, incomplete, or biased, making difficult the parameter estimation of a mixture.

In the contexts of meta-search [20] and adaptive filtering [2], the mixing parameter and the parameters of the component normal and exponential densities have been estimated without using any relevance judgements. The standard iterative *expectation maximization* (EM) method [24] was used with some success. The method can be modified to take into account relevance judgements, if any, nevertheless, it was found to be 'messy' and difficult to tune. It was very sensitive to the choice of the initial parameter values in converging to a global optimum rather than a local one.

When normalizing scores, especially of non-cooperative engines, one should keep in mind that systems produce scores in order to rank documents and do not care about the scale or shape of the scoring function. Therefore, system components which do not affect the ranking may be added or removed arbitrarily, in order to, e.g., simplify calculations. Components which affect only the scale are not a problem for mixture models. However, many transformations affect the shape as well, e.g. using a logistic function to map $(-\infty, +\infty)$ to $[0, 1]$; in such cases, the initial choice of the density components may not apply any longer.

3. SINGLE DISTRIBUTION METHODS

The analysis of Section 2 suggests that the normal-exponential mixture is not universal in modeling SDs in IR; some retrieval models perhaps could better be fitted with different mixtures, as in the case of KL-DIVERGENCE (Figure 2). Furthermore, the model has a serious theoretical problem: it does not satisfy the convexity condition, i.e. the output score does not monotonically increase with the input score. The problem shows always at the top of rankings, and it does not seem to be severe for thresholding tasks where an optimal threshold may often be lower than the non-convex 'blind' range, depending on the measure under optimization [5]. The problem is more acute in environments favoring initial precision such as in meta-search and distributed retrieval.

To make things worse, there are practical problems in estimating the parameters of mixture models, usually due to insufficient numbers of relevance judgements or quality of them (biases, incompleteness). Approaches which do not use relevance judgements seem difficult to tune, especially when relevance is sparse. Test collections are usually made in such ways that there is some minimum number of relevant items per request. In reality, given a collection, there can be no relevance for some queries. The same can happen when test collections are split further in order to facilitate distributed retrieval setups. As a result, score distributional approaches to score normalization without reference to relevance may have some merit.

[1] In practice, however, since `trec_eval` would sort on `docid` all documents with the same score, for the only purpose of preserving the order in evaluation we break the ties by multiplying each of the affected scores with $1 + \epsilon$, where ϵ is a very small number whose magnitude depends on the original score.

3.1 Z-score

A standard method for score normalization that takes the SD into account is the Z-SCORE. Scores are normalized, per topic and engine, to the number of standard deviations that they are higher (or lower) than the mean score:

$$\text{Z-SCORE:} \quad s' = \frac{s - \mu}{\delta} \tag{1}$$

where μ is the mean score and δ the standard deviation. The mean and standard deviation depend on the length of the ranking. Z-SCORE seems to assume a normal distribution of scores, where the mean would be a meaningful 'neutral' score. As it is well-known, actual SDs are highly skewed and clearly violating the assumption underlying the Z-SCORE. Although not very popular in IR, Z-SCORE was used with reasonable success in [26, 16].

3.2 Aggregate Historical CDF

A recent attempt models aggregate SDs of many requests, on per-engine basis, with single distributions [12, 13]; this enables normalization of scores to probabilities—albeit not of relevance—comparable across different engines. Nevertheless, it is not clear—if it is even possible—how using a single distribution can be applied to thresholding, where for optimizing most common measures a reference to (or probabilities of) relevance are needed. Per engine, the proposed normalization is

$$s' = F^{-1}(P(S \leq s)) \tag{2}$$

where $P(S \leq s)$ is the *cumulative density function* (CDF) of the probability distribution of all scores aggregated by submitting a number of queries to the engine, and F is the CDF of "the score distribution of an ideal scoring function that matches the ranking by actual relevance." The F^{-1} transformation is called "standardization step," it is common across all engines participating in a fusion or distributed setup, and considered critical to the method for compensating for potential individual system biases.

In a large fusion experiment using TREC Web Track data, [13] found that the method performs better than CombSUM (with standard or rank-sim normalization) and CombMNZ [18]. For score aggregation, historical queries were used, and only 25-50 seemed enough for good end-results. The method seems very promising, however, unnecessary complicated as we explain next.

3.3 Aggregate Historical CDF Simplified

By definition, F is monotonically increasing since it is a CDF. Its *quantile function* F^{-1} is also monotonically increasing, and since it is applied as a constant transformation to all engines it has no effect on rankings or the comparability of normalized scores across engines. Thus, at least in distributed retrieval setups where normalized ranked lists are simply merged, F^{-1} has no impact and it can safely be removed from the calculation. Nevertheless, it has an unclear impact and interpretation when scores are combined, e.g. in meta-search/fusion setups. The distribution in question is roughly approximated by the "average distribution of several good scoring systems", not a very well-defined concept.

Consequently, we find it hard to see why the combination of functions in Equation 2 returns the probability of relevance or any other meaningful number, and since F^{-1} is constant across engines we settle for the simpler method

$$\text{HIS:} \quad s' = P(S_{\text{HIS}} \leq s) \tag{3}$$

where HIS refers to the fact that historical queries are used for aggregating the SD that the random variable S_{HIS} follows. HIS normalizes input scores s to the probability of a historical query scoring at or below s. The aggregate historical SD is an average which can be seen as produced by an 'average' historical query. In this respect, HIS normalizes the SD of the 'average' query to uniform in $[0, 1]$. This is equivalent to the Cormack model [15], assuming such an 'average' query is sensible and exists.

4. SIGNAL-TO-NOISE METHODS

In this section we introduce a novel approach based on dual aggregate SDs but without reference to relevance. Assuming that scores produced by an engine consist of two components, signal and noise, the score random variable S can be decomposed as:

$$S = S_{\text{SIGNAL}} + S_{\text{NOISE}}$$

The probability densities of the components are given respectively by p_{SIGNAL} and p_{NOISE} defined across the engine's output score range. As we will discuss in detail in the next section, the probability densities will be estimated, per engine, by submitting appropriate sets of queries to an engine and observing the output scores.

Furthermore, we assume 'stable' system characteristics for the engine in the sense that the signal and noise levels at a score depend only on the score. We can define a function which normalizes input scores s into the fraction of the signal at s:

$$\text{S/N:} \quad s' = \frac{p_{\text{SIGNAL}}(s)}{p_{\text{SIGNAL}}(s) + p_{\text{NOISE}}(s)} \tag{4}$$

Since engines are expected to produce increasing signal-to-noise ratios as score increases, this may be an interesting normalization. Nevertheless, the magnitude of the original score is not taken into account.

An obvious improvement would be to multiply S/N with s, a procedure which would reduce s to its fraction of the signal it contains. But bare scores are usually not directly comparable across engines. As a form of calibration, we could instead use the HIS normalization of scores:

$$\text{S/N*HIS:} \quad s' = \frac{p_{\text{SIGNAL}}(s)}{p_{\text{SIGNAL}}(s) + p_{\text{NOISE}}(s)} P(S_{\text{HIS}} \leq s) \tag{5}$$

The resulting scores would be comparable across engines, however, the distribution of the variable S_{HIS} depends on some ill-defined and maybe problematic set of historical queries.

Using historical queries, although very feasible and no cooperation is required, may lead to instabilities and biases:

- Any drastic deviation of new queries from the past would imply that score transfer functions have to be re-estimated. Thus, S_{HIS} is a 'moving target'.
- If past queries hit different engines with different *generality*, engines with low average generality will produce thin S_{HIS} tails and HIS will calibrate their scores higher than other engines. This introduces a non-desirable bias; we do not see why a relevant item from a low generality engine is more relevant than others.

To deal with this, we can instead use the variable S_{SIGNAL}:

$$\text{S/N*SIG:} \quad s' = \frac{p_{\text{SIGNAL}}(s)}{p_{\text{SIGNAL}}(s) + p_{\text{NOISE}}(s)} P(S_{\text{SIGNAL}} \leq s) \tag{6}$$

The last factor calibrates s to the probability of having signal at or below s.

None of the three proposed normalizations (Equations 4-6) guarantees to preserve ranking order, while the original historical CDF and HIS (Equations 2 and 3) do. In theory, the S/N component does not have to be monotonically increasing with the score. This may

be a desirable effect—improving rankings—or not. In practice, S/N is more or less found to be monotonically increasing, at least in the current experimental setup (Section 6). Trying to enforce monotonicity in the S/N transfer functions with interpolation led to practically the same end-results, so we report without interpolation.

The question is how to approximate p_{SIGNAL} and p_{NOISE} per engine. Seeing engines as black-boxes similarly to the original historical CDF approach and HIS, we can feed each one with queries of appropriate types and generate the needed functions based on the statistical properties of the observed output scores. Next, we develop two models for generating artificial queries given a document collection. The resulting query sets may be suitable for producing aggregate SDs approximating S_{NOISE} and S_{SIGNAL}.

5. QUERY MODELS

The SDs that modern retrieval systems produce may be affected by several query characteristics. We consider three statistical features of queries:

1. Frequencies of term occurrences (e.g., Zipf, uniform).
2. Term dependencies (e.g., serial, long-range, independent).
3. Length (e.g., power-law, Poisson, etc.).

In this section, we examine these features for two query types sitting at the two extreme sides of the query spectrum: *a)* garbage (monkey/random) queries, and *b)* natural language queries.

While current query-logs show real queries consisting of bags of keywords, in any order, maybe mixed with natural language fragments (a result of the realization by users of how current systems treat their queries), this behavior is most likely to change once users realize that some system is 'understanding' natural language. The rationale for considering the two extreme types of queries is that those bounds are not going to change as systems or user behavior changes, so they provide a well-defined framework for our methods.

5.1 Monkeys on Modified Typewriters

In parallel to the popular thought experiment of a monkey hitting keys at random on a typewriter, let us imagine a keyboard with the terms of a query language on its keys plus "enter". The keys are considered equally accessible and of equal size, except "enter" which has a different size and thus different probability to be hit if keys are hit at random.

The monkey, not understanding the grammar and semantics of the query language, will select terms uniformly. Moreover, terms will be independent. Next we will examine the query length. Probabilistic foundations for the following analysis can be found in [23].

If p is the probability of hitting "enter", then the probability that the monkey will type k terms before hitting "enter" is given by (the discrete analogue of the *exponential distribution* called) the *geometric distribution*:

$$P(K_1 = k) = (1-p)^k p, \quad k = 0, 1, 2, \ldots$$

Note that a p fraction of the total queries will be of zero-length. The mean query length will be $1/p$.

Assuming r monkeys using identical keyboards (characterized by the same p) are typing independently, the random variable $K = \sum_{m=1}^{r} K_m$, where K_m is the geometrically distributed variable associated with the mth monkey, follows a *negative binomial distribution*:

$$g(k; r, p) = \binom{k+r-1}{k} p^r (1-p)^k, \quad k = 0, 1, 2, \ldots$$

Under an alternative parameterization, $\lim_{r\to\infty} g(k; r, p)$ converges to the *Poisson distribution*

$$\text{Poisson}(k; \lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$$

with a rate $\lambda = r(1/p - 1)$.

Alternatively, one may model a single monkey's query length as a Poisson distribution, i.e. as the number of terms the monkey will issue in a fixed time period t if it provides terms with a known average rate λ_1 and independently of the time passed since the last issued term. This setup may also be plausible and it involves the speed of typing and a fixed-time restriction. Nevertheless, for r monkeys producing independently Poisson distributed query lengths $K_1, K_2, \ldots K_r$ with $\lambda_1, \lambda_2 \ldots \lambda_r$, the sum K is also Poisson distributed with $\lambda = \sum_{m=1}^{r} \lambda_m$.

Consequently, the Poisson distribution is a good approximation of query lengths generated by a large number of monkeys. This query model is not dependent on the query language and it may apply to non-text retrieval as well.

5.2 Humans on Search Engines

Restricting the problem to textual data and natural language queries, it is well-known and many times confirmed that the distribution of word frequencies follows the (generalized) *Zipf's law* (Equation 7 in Section 5.2.1) with s slightly more than 1 for most types of texts. Exceptions include legal texts which have $s \approx 0.9$, showing that lawyers use more unique words than other people [14]. In any case, it seems that across natural languages and text types, s is varying a little around 1.

Query terms occur, in general, in a dependent way (i.e. the occurrence of one makes the chances of occurrence of some others better than random) due to all of them pointing at the same topic. For natural language queries, there exists also serial dependence, imposed by grammar and semantics. When incorporating dependencies, retrieval models are becoming practically intractable, which led in the past to the infamous *term independence assumption*.

Instead of trying to model term probabilities of occurrence and dependencies, we can rather tackle both features at once by picking real text fragments out of a corpus. The remaining question is how long those fragments should be.

5.2.1 A Model for Human Query Length

From analyzing query-logs, previous research has found that the distribution of query lengths can be approximated with the (generalized) *Zipf's law*[2] [27, 30]. However, the law appears to fit well to the largest query-length observations, $k \geq k_0$, but not for the whole sample, where k_0 depends on the domain. For example, empirical observations show that the length frequency for web queries peaks at 2 rather than at single keyword queries, suggesting a $k_0 > 2$. Others, without empirical justification, modeled query lengths with a Poisson distribution by setting its mean to the average query length [8].

Using a Zipf distribution, in a population of N queries the number of queries with length k is given by

$$f(k; s, N) = \frac{N}{H_{N,s}} k^{-s} \tag{7}$$

rounded to the nearest integer, where s is a positive real number and $H_{N,s}$ is the Nth *generalized harmonic number*, $H_{N,s} =$

[2]Or, in general, a *power-law*. Zipf's law can also be shown equivalent to *Pareto's distribution* by variable exchange. Essentially, the Zipf and Pareto distributions are both power-laws [1].

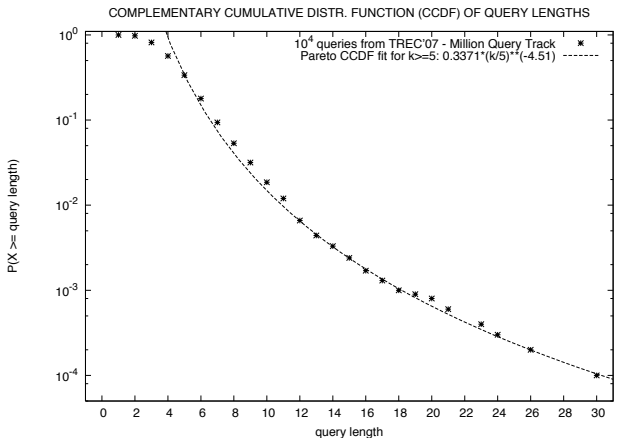

Figure 3: Pareto fit on the CCDF of query lengths.

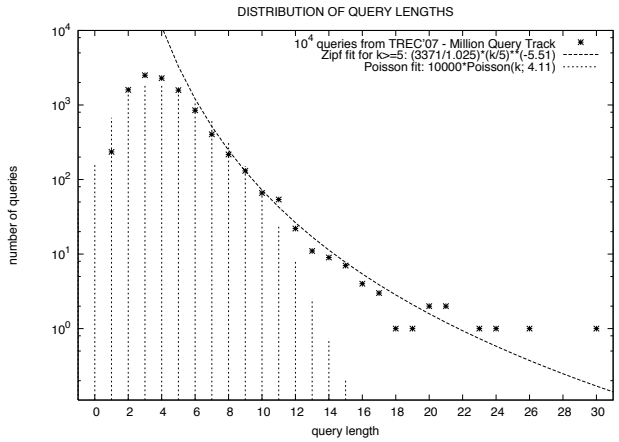

Figure 4: Zipf and Poisson fits on query lengths.

$\sum_{n=1}^{N} n^{-s}$. Although s is varying a little around 1 for word frequencies, it may be naive to extend this to query lengths.

We analyzed the query lengths of the 10k queries of TREC Million Query Track 2007. This set contains queries with lengths between 1 and 30 with a mean length of 4.11 and the peak of the distribution at length 3. We set $k_0 = 5$ (just above the mean) and estimated s for $k \geq k_0$. Rather than (logarithmically) binning the data, we instead looked at the Pareto CCDF $P(X \geq x) = (x/k_0)^{-a}$ to obtain a good fit (Figure 3). The tail naturally smoothes out in the cumulative distribution and no data is 'obscured' as in a binning procedure. This procedure is described in [1]. Fitting the Pareto CCDF, we found $a = 4.51$, consequently $s = a + 1 = 5.51$. Since this is a very good fit, we have no doubts that for $k \geq k_0$ the distribution can be modeled with a generalized Zipf-law with $s = 5.51$.

We tried the same procedure on the union of query sets (<RequestText> fields) of the TREC Legal Track 2006 and 2007 and estimated an $s' = 4.9$. Although this number is not to be trusted due to the small size of the query set (only 96 queries) which gave a very rough fit, it strengthens the evidence that the Zipf shape parameter is closer to 5 for query lengths rather than 1 (as for word frequencies). This gives a much steeper slope on log-log plots. In a further study, using 3 extra data-sets and a different—more widely-accepted—method for fitting, we have reached similar results [4].

Figure 4 shows the resulting Zipf fit on the data for $s = 5.51$ and $H_{10^4,5.51} = 1.025$. We also show (with impulses) the Poisson distribution used by [8], by setting the mean to the average query length.[3] We can see that while it matches well the data from 1 to 10, it lacks the tail for $k > 10$. The Zipf model matches better in a wider range of lengths, from 5 to longer than 20.

In summary, the bulk of queries concentrates at short lengths where a power-law does not fit at all given the current query languages, therefore it makes practical sense to use a truncated mix of Poisson-Zipf to generate query lengths. In such a practical model, the lengths are Poisson-distributed for $k < k_0$ while they are Zipf-distributed for $k \geq k_0$. The choice of k_0 depends on the specific domain (i.e., a combination of features of the document collection, query/indexing language, and pattern of use of the system). As a rule of thumb, k_0 seems to be just above the mean observed query length.

5.3 Application to the Signal-to-Noise

We expect the garbage input to generate aggregate SDs which approximate S_{NOISE}. Furthermore, generating queries by picking natural language fragments out of collection documents, some relevance is guaranteed; so we expect to approximate S_{SIGNAL} by aggregating the resulting SDs. Of course, those two aggregate SDs are bound to contain spillovers, i.e. noise contains some signal and the other way around, but they may be good enough for the signal-to-noise score normalization methods.

A disadvantage of the proposed query models in the current context is that they use the collection indexed in each engine. Therefore, they can only be applied by vendors themselves (co-operation with other engines is not needed), if there is interest to provide normalized output for further use. If there is no such interest, signal-to-noise characteristics of non-cooperative engines can still be deduced by others using alternative ways. Garbage queries could be generated in a more generic way without looking into individual collections. For approximating signal, using the internal collection seems more suitable; nevertheless, query-based sampling techniques [10] of non-cooperative engines' collections may come to rescue.

6. EVALUATION

In this section, we will experiment with the new signal-to-noise approach to score normalization. Earlier work has so far concentrated on fusion, but we will focus on distributed retrieval where the engines have no documents in common. In fusion experiments with TREC-3 and TREC-6 data, [20] found that the normal-exponential mixture performs as good as CombSUM and CombMNZ. In a large fusion experiment using TREC Web Track data [13], the historical CDF method performed better than CombSUM (with standard or rank-sim normalization) and CombMNZ [18]. Consequently, both methods perform at least as well or better than CombSUM and CombMNZ in fusion, but they have never been compared against each other. Furthermore, none of these methods has been evaluated before in distributed retrieval which is more challenging than fusing systems retrieving the same sets of documents.

6.1 Testbed

The LEMUR Toolkit[4] v4.6 was employed in three different 'flavors': *i)* TF.IDF with scores in $(0, +\infty)$, *ii)* the KL-DIVERGENCE language model with scores in $(-\infty, 0]$, and *iii)* the OKAPI proba-

[3]According to the analysis in Section 5.1, [8] used query lengths generated by r monkeys rather than humans.

[4]www.lemurproject.org

Table 1: Relative performance of 3 retrieval models on TREC-123 (topics 51-150) and TREC-4 (topics 201-250).

TREC-123

run	P5	P10	P15	P20	P30	MAP
TF.IDF	0.5160	0.4800	0.4727	0.4505	0.4323	0.1816
KL-DIV	0.4980	0.4590	0.4347	0.4345	0.4157	0.1646
OKAPI	0.4820	0.4730	0.4493	0.4405	0.4217	0.1602

TREC-4

run	P5	P10	P15	P20	P30	MAP
TF.IDF	0.4760	0.4220	0.3827	0.3660	0.3247	0.2125
KL-DIV	0.4680	0.4260	0.4000	0.3700	0.3407	0.2036
OKAPI	0.5280	0.4420	0.4053	0.3730	0.3420	0.2070

bilistic model with scores in $(-\infty, +\infty)$. [5] The default parameters that come with this version of the toolkit were used; we just enabled Porter stemming and the SMART stop-word list in indexing. To test the setup, we did runs on a standard index with all the documents, using each of the three 'flavors.' Table 1 presents the results of the three retrieval models, and may serve as a performance ceiling for the distributed retrieval experiment. All three models perform reasonably at MAP and early precision, making our experimental results representative for a large group of systems.

We used two collections, the TREC-123 and TREC-4 data, each one partitioned in 100 non-overlapping sub-collections according to the CMU split [10]. The TREC-123 split is by document source and relevant documents appear scattered throughout all the sub-collections. The TREC-4 split is topically clustered and relevant documents appear in relatively few sub-collections. To build the test engines, we applied the three retrieval models to the sub-collections in a round-robin fashion, i.e., TF.IDF / KL-DIV / OKAPI / TF.IDF / ...

As queries, we used the TREC topics corresponding to the collections, i.e. 51-150 for TREC-123 and 201-250 for TREC-4. We only processed the <desc> field; this is the only field that the extended set of topics 51-250 have in common. We evaluate on 100 and 50 queries respectively but we use topics 151-200 as historical data as we will see below.

Most of the normalization methods we compared are based on observed SDs for which data are sparse at the top scores/ranks. Thus, following common practice in distributed retrieval, we found most appropriate to evaluate with average precision at ranks 5, 10, 15, 20, and 30.

We were also interested in how the methods interact with resource selection. We employed the standard resource selection method of LEMUR, CORI [9], to select only the top-10 sub-collections per query.

6.2 Setting the Baseline

Our approach is based on the assumption that effective score normalization needs to take the SD into account. Before trying the new normalization methods, we will investigate whether our assumption indeed holds.

Standard score normalization methods like the MinMax ignore the score distribution: $s' = \frac{s-min}{max-min}$, with min (max) the minimal (maximal) score per query and engine [17]. That is, MinMax forces all scores in [0,1], resulting in a maximal score per topic and engine of 1. In our distributed retrieval set-up, as a result, the first 100 results per topic will have the maximal score of 1, and we will be doing effectively a round-robin picking the top result of each engine. Hence, we will look at ROUNDROBIN to represent the familiar score normalization methods that ignore the shape of the SD.

[5]We also intended to use INQUERY in order to have as diverse models as possible, but we discovered a bug in its implementation in the current LEMUR version. The bug most likely does not affect ad-hoc retrieval performance, but it affects tasks where score normalization is needed.

Table 2: Distributed retrieval results for TREC-123 and TREC-4 over all 100 engines using earlier score normalization methods. Significant-tested with a bootstrap test, one-tailed, at significance levels 0.05 ($^\circ$), 0.01 ($^\circledcirc$), 0.001 ($^\bullet$).

TREC-123

run	P5	P10	P15	P20	P30
ROUNDROBIN	0.1835	0.1835	0.1835	0.1835	0.1835
Z-SCORE 250	$0.2340^{\circledcirc}$	0.2150°	$0.2220^{\circledcirc}$	$0.2260^{\circledcirc}$	$0.2207^{\bullet}$
Z-SCORE 1K	0.2180^{-}	$0.2320^{\circledcirc}$	$0.2320^{\circledcirc}$	$0.2285^{\circledcirc}$	$0.2167^{\circledcirc}$
HIS	$0.2480^{\circledcirc}$	0.2340°	0.2193°	0.2120^{-}	0.2017^{-}

TREC-4

run	P5	P10	P15	P20	P30
ROUNDROBIN	0.0584	0.0584	0.0584	0.0584	0.0584
Z-SCORE 250	$0.1360^{\bullet}$	$0.1360^{\bullet}$	$0.1320^{\bullet}$	$0.1200^{\bullet}$	$0.1053^{\bullet}$
Z-SCORE 1K	$0.1480^{\bullet}$	$0.1300^{\bullet}$	$0.1253^{\bullet}$	$0.1130^{\bullet}$	$0.0940^{\circledcirc}$
HIS	$0.2280^{\bullet}$	$0.1920^{\bullet}$	$0.1627^{\bullet}$	$0.1540^{\bullet}$	$0.1487^{\bullet}$

The effectiveness of ROUNDROBIN is strongly dependent on the order of the engines, and given their disjoint collections, the precision up to rank 100 depends *only* on their order. To avoid this arbitrariness, we calculated the average precision over all possible orderings of the engines; this turns out to be equal to P100 for all ranks down to 100.[6] The results of ROUNDROBIN are shown in Table 2. It performs poorly on the topically clustered TREC-4, and somewhat better on TREC-123. Although MinMax is one of the most obvious baseline methods for score normalization, it is clear that it is a weak baseline with flat precision up to rank 100 in our set-up.

We calculate also the Z-SCORE over the top 1,000 results, and over the top 250 results per query. The results of Z-SCORE are also shown in Table 2, and it performs significantly better than ROUNDROBIN. This is a strong argument in favor of taking the SD into account. Even more so given that Z-SCORE seems to assume a normal distribution of scores, where the mean would be a meaningful 'neutral' score. As we have seen before, actual SDs are highly skewed and clearly violating the assumptions underlying the Z-SCORE. Hence, we would expect even better performance from a method that is tailored to the sort of SDs we deal with in IR.

As we will see below, the normal-exponential model gave poor fits on sub-collections assigned the KL-DIVERGENCE model and on some with few relevant documents; thus, it would have made a weak baseline. The historical CDF approach [13] we discussed in Section 3 is one of the latest well-performing distributional methods in the literature. It derives a transfer function from historical data (hence we call its simplified version without the extraneous

[6]The average precision over n engines with disjoint content, when averaged over all possible orderings, is equal to Pn for all ranks down to n. We omit the full proof but only calculate the Pn and P1: Suppose that m ($0 \leq m \leq n$) engines have a relevant top result. No matter how the engines are ordered, at rank n we will have selected precisely the top result of all n engines, and hence Pn will be m/n. What will the P1 be? In exactly m of the choices of the first engine, P1 will be 1, and in the remaining $n - m$ choices of the first engine, P1 will be 0. Hence, averaging over all possible orderings, average P1 will also be m/n. The proof for the intermediate ranks is more elaborate, but follows similar arguments, and is omitted.

standardization step—as we argued in Section 3.3—HIS), and its effectiveness is shown in Table 2. As it turns out, HIS is also significantly better than ROUNDROBIN. On TREC-123, HIS is similarly effective to Z-SCORE, and on the topically clustered TREC-4 it is clearly better than Z-SCORE. HIS achieves roughly 50% of the precision obtained on the full index in Table 1. Hence, we will use HIS as a strong baseline for the experiments with the signal-to-noise approach.

6.3 Runs

We compared HIS against the older normal-exponential normalization NORMEXP, and the new signal-to-noise methods S/N, S/N∗HIS, and S/N∗SIG.

As historical queries we used the <desc> fields of the remaining TREC topics 151-200 in order to *a)* construct the transfer functions for the HIS runs, and *b)* set the average query length parameter of both artificial query models to $\lambda = 23.3$ and $k_0 = 24$. [13] reported good end-results even with only 25-50 historical queries, and since we only have 50 historical, we use 50 artificial ones for the signal-to-noise runs per query type (i.e. 50 for signal and 50 for noise).

Figure 5 shows the high-ends of some typical transfer functions. For illustration we randomly selected 3 engines, one for each retrieval model, and depict the log-odds of the output scores since they are very close to 1. The calculations are limited by the machine and software arithmetic precision at around 36.74 log-odds, where the transfer functions flatten out. The extend of the non-flat areas were deemed sufficient for the distributed retrieval setup for three out of the four methods; only S/N moves to too large numbers sometimes a bit too early, i.e. before the top score of a run is reached. We employed *kernel density estimation* techniques [28] for estimating the required probability densities from data.

Concerning the NORMEXP run, using EM for parameter estimation without relevance judgements or any indication of where the component densities may lie, is a very difficult task. We tried fitting all scores in $[s_{\text{peak}}, +\infty]$ but the reasonable fits did not capitalize in end-results; fitting, e.g., on a fixed top-1,000 gave better retrieval precision. We tried numerous initial settings in EM, but no settings seemed universal. While some settings helped a lot some queries, they had a negative impact on others. After a few cycles of optimizing-evaluating, we obtained the best end-results for the following settings: fit on the top-100 scores with initial parameter values for the densities (i.e. the means of the normal and exponential, and standard deviation of the normal) bootstrapped randomly in the interval (s_{100}, s_1) and a mixing parameter in $(0, 1)$. The random initial parameters approach worked better than looking into specific ranges, because we run EM ten times and selected the fit with the least square error with the score data.

We did two batches of runs: with and without resource selection. Without resource selection, retrieval was performed on all sub-collections. All document scores per engine were passed through the engine's transfer function, and the resulting ranked lists from all engines were simply merged. For the runs with resource selection, we merged only the results of the top-10 sub-collections as selected by CORI.

6.4 Results and Discussion

Table 3 presents the distributed retrieval results without resource selection. Compared to the most similar—but not directly comparable—setup in the literature, namely that of [21] on TREC-123 only, our precision seems to be lower. Our setup is more challenging than the last-cited study due to the use of 3 different retrieval models assigned in a round-robin fashion over the 100 sub-collections rather

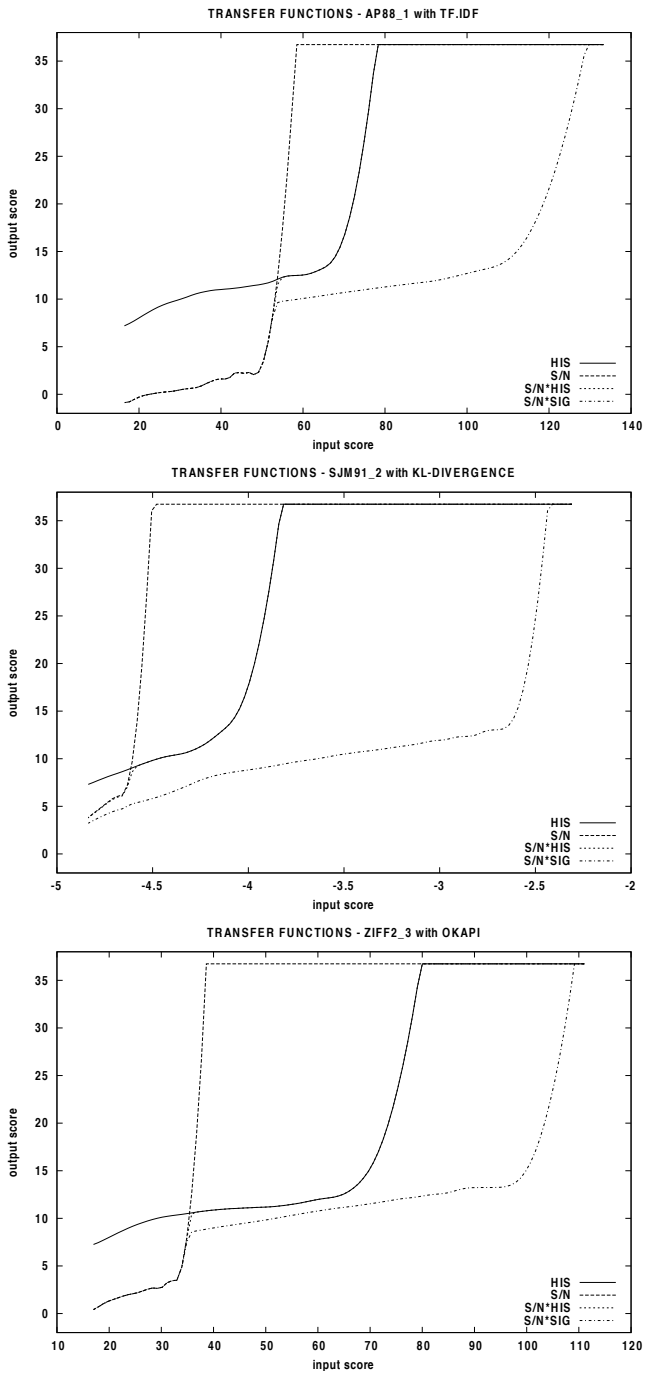

Figure 5: Score transfer functions for a) TF.IDF, b) KL-DIVERGENCE, and c) OKAPI. The 'hidden' function is the S/N∗HIS curve, which overlaps with S/N and S/N∗SIG at low input scores and with HIS elsewhere.

than just a single model. Furthermore, we did not tune anything on the collection but used the default LEMUR settings.

Overall, the S/N∗HIS and S/N∗SIG runs show significant improvements over the strong baseline of HIS, while the consistent improvements in S/N are mostly non-significant. S/N∗SIG obtains roughly 70% of the scores on the full index in Table 1. NORMEXP is disappointing across the board, especially for the topically clustered TREC-4, and it does even worse than ROUNDROBIN shown in Table 2.

Table 3: Distributed retrieval results for TREC-123 and TREC-4 over all 100 engines. Significant-tested with a bootstrap test, one-tailed, at significance levels 0.05 (°), 0.01 (°), 0.001 (•).

TREC-123

run	P5	P10	P15	P20	P30
HIS	0.2500	0.2400	0.2240	0.2165	0.2047
NORMEXP	0.1700$^{\circ\circ}$	0.1590$^{\circ\circ}$	0.1440$^{\bullet}$	0.1375$^{\bullet}$	0.1323$^{\bullet}$
S/N	0.2620^{-}	0.2630^{-}	0.2533^{-}	0.2495°	0.2290^{-}
S/N∗HIS	0.3200$^{\bullet}$	0.3020$^{\bullet}$	0.2860$^{\bullet}$	0.2770$^{\bullet}$	0.2537$^{\bullet}$
S/N∗SIG	0.3680$^{\bullet}$	0.3380$^{\bullet}$	0.3180$^{\bullet}$	0.3095$^{\bullet}$	0.2790$^{\bullet}$

TREC-4

run	P5	P10	P15	P20	P30
HIS	0.2280	0.1920	0.1627	0.1540	0.1487
NORMEXP	0.0840$^{\bullet}$	0.0780$^{\bullet}$	0.0693$^{\bullet}$	0.0600$^{\bullet}$	0.0487$^{\bullet}$
S/N	0.2080^{-}	0.1980^{-}	0.1800^{-}	0.1740^{-}	0.1560^{-}
S/N∗HIS	0.2720°	0.2380$^{\circ\circ}$	0.2067$^{\circ\circ}$	0.1920$^{\circ\circ}$	0.1740°
S/N∗SIG	0.2600^{-}	0.2400°	0.2173$^{\circ\circ}$	0.2090$^{\bullet}$	0.1793°

Table 4: Distributed retrieval results for TREC-123 and TREC-4 over the 10 engines selected by CORI resource selection. Significant-tested with a bootstrap test, one-tailed, at significance levels 0.05 (°), 0.01 (°), 0.001 (•).

TREC-123

run	P5	P10	P15	P20	P30
HIS	0.3280	0.3030	0.2827	0.2630	0.2283
NORMEXP	0.3220^{-}	0.2920^{-}	0.2747^{-}	0.2595^{-}	0.2280^{-}
S/N	0.3060^{-}	0.2830^{-}	0.2553°	0.2460^{-}	0.2160^{-}
S/N∗HIS	0.3420^{-}	0.3060^{-}	0.2773^{-}	0.2590^{-}	0.2223^{-}
S/N∗SIG	0.3540^{-}	0.3120^{-}	0.2740^{-}	0.2580^{-}	0.2217^{-}

TREC-4

run	P5	P10	P15	P20	P30
HIS	0.2840	0.2560	0.2320	0.2040	0.1800
NORMEXP	0.2400^{-}	0.1980°	0.1720°	0.1540°	0.1353°
S/N	0.2440^{-}	0.2320^{-}	0.2160^{-}	0.1950^{-}	0.1747^{-}
S/N∗HIS	0.2960^{-}	0.2680^{-}	0.2320^{-}	0.2060^{-}	0.1793^{-}
S/N∗SIG	0.3000^{-}	0.2760^{-}	0.2480^{-}	0.2170^{-}	0.1827^{-}

Table 4 presents the distributed retrieval results with resource selection. Here, HIS performs substantially better than in Table 3 above. Also NORMEXP now performs very close to the baseline of HIS in TREC-123, but it still loses in TREC-4. Overall, S/N tends to be less effective than HIS, but the differences are mostly insignificant. S/N∗HIS and S/N∗SIG are most of the time better than HIS but also here the differences are insignificant.

While NORMEXP is disappointing in the distributed experiment, it works well in fusion, as other studies have found in the past. Further examination revealed two main reasons for not performing well in this particular distributed retrieval experiment. First, some sub-collections have very few or no relevant documents. In these cases, Gaussian fittings became spurious, hurting end-performance. Resource selection effectively eliminates most of this problem. Second, the KL-DIVERGENCE, as shown in Figure 2, was a problematic model fitting-wise. In the current setup, we obtained reasonable normal-exponential fits on the SDs produced by OKAPI and TF.IDF, with the latter giving the best fits.

Remarkably, while S/N and S/N∗HIS are also profiting from CORI, S/N∗SIG tends to perform worse on the 10 selected engines than on all engines in Table 3. Apparently, this normalization does not need resource selection, or in other words it takes care of resource selection by assigning the higher scores to the engines containing the relevant documents. Of course, resource selection may be needed in practice for efficiency reasons.

7. CONCLUSIONS

We investigated new methods for score normalization, a fundamental IR problem affecting all settings where results from different search engines have to be compared or merged. The approach taken deduces normalization functions from the observed aggregate SDs for statistically well-defined query inputs. For this purpose, we have developed two concrete input query models, one for natural language fragments and the other for garbage/noise, dealing not only with the choice of keywords but also with query length. These query models produce SDs which are supposed to approximate the score signal and noise of each engine, but—despite the positive end-results—they may not be the best for the purpose. Further research into more suitable input query models may lead to improved performance.

The proposed normalization methods characterize each engine with a single score transfer function. The implicit assumption made is that engines provide 'stable' outputs, in the sense that scores produced by a given engine for different queries are at least comparable with each other. In this respect, we generated SDs for 'average' queries. Nevertheless, if the SDs of a retrieval model are affected greatly by some specific query feature, e.g. the sum of IDFs, query length, etc., then profiling could be done in ranges of values of such a feature. The methods may be computationally expensive but practically feasible and efficient, since transfer functions can be pre-calculated offline and may only have to change with significant collection updates.

We conducted a series of experiments trying to establish the practical utility of the resulting normalization methods for merging rankings in a distributed retrieval setup. In a first set of experiments we compared existing methods of score normalization: *i)* round-robin corresponding to score normalizations that maps the highest score to 1, *ii)* z-score as an non-IR normalization method that takes the distribution into account, and *iii)* a cumulative density function (CDF) derived from aggregating scores resulted from a set of historical queries. The results clearly showed that taking the shape of the SD into account leads to significantly better performance. We decided to take the historical CDF approach, performing at roughly 50% of the non-distributed runs, as a strong baseline for further experiments.

The second set of experiments investigated the effectiveness of the signal-to-noise approaches, in relation to the earlier normal-exponential model and to the historical CDF method. The signal-to-noise methods led to significantly better performance than the historical CDF method, performing as high as 70% of the non-distributed runs. We also compared against the normal-exponential mixture model for score normalization—its first evaluation in a distributed setup as far as we are concerned—whose performance did not live up to its underlying theory. The review of the normal-exponential should serve as a starting point for improving mixture SD models.

Any mixture model would be bound by sparse, incomplete, or biased relevance, making the estimation of the component densities difficult. The sparsity weakness showed up in our distributed retrieval experiments where some sub-collections had few or no relevant documents. Where enough relevant documents exist, or when some relevance judgements are given, the model has performed well as shown in previous studies. In addition, its lack of universality affected greatly our results: the mixture did not fit well in one of the three retrieval models we used, namely, the KL-divergence. How we dealt with the non-convexity may have also played a role;

effectiveness may vary across different ways of fixing the problem, especially in tasks favoring initial precision.

The third set of experiments looked at the effectiveness of the above methods in combination with resource selection (RS). With RS, the signal-to-noise methods still improve over the historical CDF method, but the improvement is no longer significant. Where the RS leads to substantial better scores for the historical CDF method, it fails to improve for the signal-to-noise approach. This can be interpreted as a positive sign. In an ideal normalization, e.g. where scores are normalized to probabilities of relevance, any kind of RS will hurt effectiveness. In such an ideal situation, the more systems one combines, the better the effectiveness. The fact that RS no longer improves the effectiveness of distributed retrieval suggests that we are breaking a 'theoretical ceiling' in normalization methods. The ultimate goal is to perform as well as the non-distributed index.

8. ACKNOWLEDGMENTS

This research was supported by the Netherlands Organization for Scientific Research (NWO), CATCH programme, under project number 640.001.501. We thank Nir Nussbaum (University of Amsterdam) for the programming support, George Paltoglou (University of Macedonia) for the technical advice, and Marijn Koolen (University of Amsterdam) for participating in the initial valuable discussions.

9. REFERENCES

[1] L. Adamic and B. Huberman. Zipf's law and the internet. *Glottometrics*, 3(1):143–150, 2002.

[2] A. Arampatzis. Unbiased s-d threshold optimization, initial query degradation, decay, and incrementality, for adaptive document filtering. In *TREC*, 2001.

[3] A. Arampatzis, J. Beney, C. H. A. Koster, and T. P. van der Weide. Incrementality, half-life, and threshold optimization for adaptive document filtering. In *TREC*, 2000.

[4] A. Arampatzis and J. Kamps. A study of query length. In *Proceedings SIGIR'08*, pages 811–812. ACM, 2008.

[5] A. Arampatzis, J. Kamps, and S. Robertson. Where to stop reading a ranked list? Threshold optimization using truncated score distributions. In *Proceedings SIGIR'09*, pages 524–531. ACM, 2009.

[6] A. Arampatzis, S. Robertson, and J. Kamps. Score distributions in information retrieval. In *Proceedings of the 2nd International Conference on Theory of Information Retrieval (ICTIR'09)*, pages 139–151, Microsoft Research, Cambridge, UK, September 2009.

[7] A. Arampatzis and A. van Hameren. The score-distributional threshold optimization for adaptive binary classification tasks. In *Proceedings SIGIR'01*, pages 285–293. ACM, 2001.

[8] L. Azzopardi, M. de Rijke, and K. Balog. Building simulated queries for known-item topics: an analysis using six european languages. In *Proceedings SIGIR'07*, pages 455–462. ACM, 2007.

[9] J. P. Callan. *Distributed Information Retrieval*, chapter 5, pages 127–150. Kluwer Academic Publishers, 2000.

[10] J. P. Callan and M. E. Connell. Query-based sampling of text databases. *ACM Transactions on Information Systems*, 19(2):97–130, 2001.

[11] K. Collins-Thompson, P. Ogilvie, Y. Zhang, and J. Callan. Information filtering, novelty detection, and named-page finding. In *TREC*, 2002.

[12] M. Fernández, D. Vallet, and P. Castells. Probabilistic score normalization for rank aggregation. In *ECIR'06*, volume 3936 of *Lecture Notes in Computer Science*, pages 553–556. Springer, 2006.

[13] M. Fernández, D. Vallet, and P. Castells. Using historical data to enhance rank aggregation. In *Proceedings SIGIR'06*, pages 643–644. ACM, 2006.

[14] L. Q. Ha, E. I. Sicilia-Garcia, J. Ming, and F. J. Smith. Extension of Zipf's law to words and phrases. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 1–6. ACL, 2002.

[15] D. Hawking and S. Robertson. On collection size and retrieval effectiveness. *Inf. Retr.*, 6(1):99–105, 2003.

[16] J. Kamps, M. de Rijke, and B. Sigurbjörnsson. Combination methods for crosslingual web retrieval. In *CLEF'05*, volume 4022 of *Lecture Notes in Computer Science*, pages 856–864. Springer, 2006.

[17] J. H. Lee. Combining multiple evidence from different properties of weighting schemes. In *Proceedings SIGIR'95*, pages 180–188. ACM, 1995.

[18] J. H. Lee. Analyses of multiple evidence combination. In *Proceedings SIGIR'97*, pages 267–276. ACM, 1997.

[19] D. D. Lewis. Evaluating and optimizing autonomous text classification systems. In *Proceedings SIGIR'95*, pages 246–254. ACM, 1995.

[20] R. Manmatha, T. M. Rath, and F. Feng. Modeling score distributions for combining the outputs of search engines. In *Proceedings SIGIR'01*, pages 267–275. ACM, 2001.

[21] H. Nottelmann and N. Fuhr. From uncertain inference to probability of relevance for advanced IR applications. In *Proceedings of the 25th European Conference on Information Retrieval Research (ECIR'03)*, LNCS 2633, pages 235–250, 2003.

[22] D. W. Oard, B. Hedin, S. Tomlinson, and J. R. Baron. Overview of the TREC 2008 legal track. In *TREC*, 2008.

[23] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, 2nd edition, 1984.

[24] B. D. Ripley and N. L. Hjort. *Pattern Recognition and Neural Networks*. Cambridge University Press, New York, NY, USA, 1995.

[25] S. Robertson. On score distributions and relevance. In *Proceedings of 29th European Conference on IR Research (ECIR'07)*, pages 40–51. Springer, 2007.

[26] J. Savoy. Report on CLEF-2003 multilingual tracks. In *CLEF'03*, volume 3237 of *Lecture Notes in Computer Science*, pages 64–73. Springer, 2004.

[27] S. Sharma, L. T. Nguyen, and D. Jia. Ir-wire: A research tool for p2p information retrieval. In *SIGIR Open Source Workshop*, Seattle, 2006. ACM.

[28] L. Wasserman. *All of Statistics: A Concise Course in Statistical Inference (Springer Texts in Statistics)*. Springer, September 2004.

[29] Y. Zhang and J. Callan. Maximum likelihood estimation for filtering thresholds. In *Proceedings SIGIR'01*, pages 294–302. ACM, 2001.

[30] I. Zukerman and E. Horvitz. Using machine learning techniques to interpret wh-questions. In *ACL'01: Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 547–554. ACL, 2001.