Beating the Bronze Medalist: Limited Merging of Disjoint Rankings

R. Lempel S. Moran
Department of Computer Science
The Technion,
Haifa 32000, Israel
email: {rlempel,moran}@cs.technion.ac.il

April 6, 2003

Abstract

Large-scale Web search engines currently index billions of Web documents. These enormous indices are often distributed across multiple index segment. One common architecture partitions the document set into several disjoint subsets, indexing each subset on a separate machine (index segment). When answering queries, the architecture requires executing each query across all segments, and then merging the results from all segments prior to returning them to the user.

We study several combinatorial issues pertaining to the number of results that should be retrieved from each segment during query execution. This number depends on (1) the number of results required by the user, (2) the number of index segments, and (3) a probabilistic quality policy employed by the search engine. We present procedures for exact (non-asymptotical) computation of this number, along with online procedures that estimate the quality of the merged results that are returned to the users.

1 Introduction

It is late August, 2004, at the Athens Olympic Sports Complex. The eight finalists of the men’s Olympic 100 meter dash, representing five different countries, set off at the sound of the gun. Less than 10 seconds later, separated only by the photo-finish camera, the gold and silver medalists, both representing the same country, are decided. A fraction of a second later, the bronze medalist, who is not a compatriot of the faster duo, crosses the finish line. The gold medalist is immediately crowned by the media, televising the event all over the globe, as the world’s fastest man. The silver medalist is proclaimed as the second fastest. Can the bronze medalist claim to be the third fastest man on the planet? The answer to this question depends on the number of entrants allowed from each country in the Games. If no country can be represented by more than two sprinters, it may very well be that the fastest compatriot of the gold and silver medalists is faster than the bronze medalist.

Formally, this paper studies the following scenario, taken from the domain of Web search engines. Consider a set of documents \( D = \{d_1, d_2, \ldots, d_N\} \), where each document is assigned randomly,
uniformly and independently into one of \( m \) index segments. We will refer to this random process as the \textit{indexing} of the documents. Then, a \textit{query} \( q \in Q \), from some set \( Q \), arrives. Each document \( d \in D \) attains a score \( s_d(q) \) with respect to the query \( q \), where \( s_d(q) \overset{\Delta}{=} f(d, q) \) for some \textit{scoring} function \( f : D \times Q \rightarrow \mathbb{R} \). These scores reflect the \textit{relevance} of each document to the query. The mission is to \textit{retrieve} the \( n \) most \( q \)-relevant documents, where \( D \gg n \cdot m \). The retrieval process is handled by a central machine called the \textit{query integrator (QI)}, in the following manner:

- The QI sends \( q \) to each of the \( m \) segments, requesting each one to return its top \( \ell \) matches (the \( \ell \) most \( q \)-relevant documents assigned to that segment).
- The QI merges the \( m \) lists of size \( \ell \), returning the top-\( n \) merged results to the user who submitted \( q \). These \( n \) results are called the \textit{result set}.

It is assumed that the QI has no knowledge on the outcome of the indexing process.

Obviously, for the above retrieval process to be well defined, \( \ell \geq \lceil \frac{n}{m} \rceil \). However, in order to ensure that the result set indeed contains the \( n \) most relevant documents with respect to \( q \), \( \ell \) must equal \( n \), since there is a positive probability that the top-\( n \) \( q \)-relevant documents were all randomly assigned to the same segment. Note that the value of \( \ell \) impacts the processing time in each segment, the communication between the QI and each of the \( m \) segments and the storage requirements of the QI. Thus, a practical implementation might set \( \ell \) to some value between \( \lceil \frac{n}{m} \rceil \) and \( n \), reducing the computations required for this process, while maintaining a high probability of receiving high-quality results - all or most of the top-\( n \) documents per query.

The following sections investigate several aspects of this problem:

- Section 2 presents a preliminary result from [13] which calculates the smallest (a-priori) value of \( \ell \), so that the result set will contain all of the top-\( n \) documents with probability no less than \( p \).

- Given a parameter \( k \) \((k < n)\), Section 3 calculates the smallest (a-priori) value of \( \ell \), so that the result set will contain at least \( n - k \) of the top-\( n \) documents with probability \( \geq p \).

- When \( \ell \) is set as calculated in Section 2 with respect to a probability \( p < 1 \), there is a chance of about \( 1 - p \) that the result set will not live up to expectations and not contain the top-\( n \) \( q \)-relevant results. In such cases, the QI can estimate, during the process of merging the \( \ell m \) results returned by the segments, several metrics on the sub-optimality of the result set. These metrics and the estimation process are the topic of Section 4.

Section 5 provides additional background on the practical setting of the problem in large-scale Web search engines. Related work concerning the mathematical aspects of this paper is cited in Section 6.

2 Retrieving the Top-\( n \) Documents for a Query

Every query \( q \) implies the identities of the \( n \) most relevant documents that pertain to \( q \). Let \( l_p(n, m) \) denote the smallest number \( \ell \) of documents that should be retrieved by the QI from each of the \( m \) segments, so that the result set contains the \( n \) most relevant documents for a query with probability \( \geq p \) \((0 < p < 1)\). By the randomness of the indexing process, \( l_p(n, m) \) is independent of \( q \). This section brings recursive formulae with which \( l_p(n, m) \) can be calculated in a time that is polynomial in \( nm \).
We start by presenting a rough upper bound on \( l_p(n, m) \). Clearly, \( \lfloor \frac{n}{m} \rfloor \) is a lower bound on \( l_p(n, m) \) for all \( p > 0 \). We show that \( l_p(n, m) \) need not be larger than
\[
\max \{ \lfloor \lambda + \log m \rfloor, \lfloor \frac{2en}{m} \rfloor \}
\]
where \( \lambda \triangleq -\log_2(1-p) \) (i.e., \( p = 1 - \frac{1}{2^\lambda} \)).

The probability that exactly \( i \) of the \( n \) results are inserted into a given segment is \( \binom{n}{i} \left( \frac{1}{m} \right)^i \left( 1 - \frac{1}{m} \right)^{n-i} \).

Since \( \binom{n}{i} \leq \left( \frac{ne}{i} \right)^i \) [17],
\[
\binom{n}{i} \left( \frac{1}{m} \right)^i \left( 1 - \frac{1}{m} \right)^{n-i} < \binom{n}{i} \left( \frac{1}{m} \right)^i \left( \frac{ne}{i} \right)^i = \left( \frac{ne}{m} \right)^i.
\]

Hence, the probability that more than \( \ell \) results are inserted into a given segment is bounded by
\[
\sum_{i=\ell+1}^{n} \left( \frac{ne}{m} \right)^i < \sum_{i=\ell+1}^{\infty} \left( \frac{ne}{m} \right)^i = \frac{\left( \frac{ne}{m} \right)^\ell}{1 - \frac{ne}{m}}.
\]

Whenever \( \ell \geq \max \{ \lfloor \lambda + \log m \rfloor, \lfloor \frac{2en}{m} \rfloor \} \), this last expression is bounded from above by \( \left( \frac{\lambda}{2} \right)^{\lambda + \log m} = \frac{1}{m^{2^\lambda}} \). Thus, by the union bound, the probability that at least one of the \( m \) segments contains more than \( \ell \) results is smaller than \( \frac{1}{m^\lambda} \). The result follows.

We now turn to the precise calculation of \( l_p(n, m) \). Given a query \( q \), we can consider the indexing process described in the Introduction as the following random process: \( n \) different balls (the top results for \( q \)) are thrown randomly, uniformly and independently into \( m \) different cells (the segments), where \( n_i \) balls are inserted to cell \( i \) (i.e., \( \sum_{i=1}^{m} n_i = n \)). Now, when a value of \( \ell \) is set, we can look at the QI as removing \( \min \{ \ell, n_i \} \) balls from cell \( i \) for \( i = 1, \ldots, m \). Denote by \( e_{n,m,\ell} \) the number of excess balls that remain in the cells after the removals of the balls. With this notation, we wish to calculate the probability \( Pr[e_{n,m,\ell} = 0] \), corresponding to the case where no cell contains more than \( \ell \) balls, so that the QI indeed managed to collect the top \( n \) results from the segments.

The number of ways to throw \( n \) different balls into \( m \) different cells is \( m^n \). We will actually be counting \( N(n, m, \ell) \), the number of ways to throw \( n \) different balls into \( m \) different cells so that no cell contains more than \( \ell \) balls, and then
\[
Pr[e_{n,m,\ell} = 0] = N(n, m, \ell)/m^n.
\]

The following recursive formulae may be used to calculate the \( N(n, m, \ell) \) values:
\[
N(n+1, m, \ell) = m \cdot \sum_{j=0}^{\ell-1} \binom{n}{j} N(n-j, m-1, \ell)
\]
\[
N(n, m+1, \ell) = \sum_{j=0}^{\ell} \binom{n}{j} N(n-j, m, \ell)
\]
\[
N(n, m, \ell) = \sum_{j=0}^{\lfloor \frac{\ell}{2} \rfloor} \binom{m}{j} \binom{\ell, \ldots, \ell, n-j\ell}{j, \ldots, j} N(n-j\ell, m-j, \ell-1)
\]

\(^1\text{Sharper asymptotic bounds on } l_p(n, m) \text{ are discussed in Section 6.}\)
The first formula represents choosing one of \( m \) cells for ball number \( n + 1 \), and then putting some number \( j \), \( 0 \leq j \leq \ell - 1 \) of additional balls into the same cell. The rest of the \( n - j \) balls then need to be put into \( m - 1 \) cells, respecting the limit of \( \ell \) balls per cell. The second formula represents choosing \( j \), \( 0 \leq j \leq \ell \) balls to populate cell \( m + 1 \), and distributing the remaining \( n - j \) balls into \( m \) cells, respecting the limit of \( \ell \) balls per cell. In the third formula, we first choose some \( j \) cells to have exactly \( \ell \) balls. We then choose the balls to populate those cells (the multinomial coefficient has \( j \) \( \ell \)-terms). The remaining \( n - j\ell \) balls are distributed to the remaining \( m - j \) cells, with each such cell collecting no more than \( \ell - 1 \) balls. All three formulae use the following initial values:

1. For all \( m, \ell, N(0, m, \ell) = 1 \). Whenever \( n > 0 \), \( N(n, 0, \ell) = N(n, m, 0) = 0 \).
2. For all \( n > 0, m > 0 \):
   - Whenever \( \ell < \lceil \frac{n}{m} \rceil \), \( N(n, m, \ell) = 0 \). This implies that the sum over \( j \) in the third recursive formula contains no more than \( 1 + m \) terms.
   - \( N(n, m, \lceil \frac{n}{m} \rceil) = \binom{m}{k} \frac{n!}{\lceil \frac{n}{m} \rceil^m} \), where \( k \triangleq n \mod m \).

Using standard dynamic programming techniques, \( N(n, m, \ell) \) can be calculated in \( O(nm^2\ell) = O(n^2m^2) \). Furthermore, when \( p \) is constant and \( n \) grows, Equation 1 asserts that \( \ell p(n, m) \leq 6n \) and so \( O(nm^2p(n, m)(m + 1)) = O(n^2m^2) \).

### 3 Retrieving Most of the Top Results

Consider next the QI’s willingness to produce less than optimal results, allowing at most \( k \) of the top \( n \) results not to be retrieved by the QI. This reflects a more relaxed quality-related policy of the QI, requiring that with high probability, most (but not necessarily all) of the top results are returned to the user. Using the balls-into-cells terminology of the previous section, we are allowing some cells to receive more than \( \ell \) balls each (thus losing the excess results), as long as the total excess lost does not exceed \( k \). Our goal in this subsection is to calculate \( Pr[e_{n,m,\ell} \leq k] \).

Denote the number of balls that were thrown into cell \( i \) by \( n_i \), and let \( f \) denote the number of overflowing cells (the number of cells into which more than \( \ell \) balls were thrown):

\[
f \triangleq |\{n_i, 1 \leq i \leq n : n_i > \ell\}|
\]

How many balls were actually thrown into these \( f \) cells? Certainly we must allow at least \( f(\ell + 1) \) balls to be assigned to these \( f \) cells, in order to cause their overflow. However, we cannot assign more than \( f\ell + k \) balls to the overflowing cells, otherwise the excess will surely exceed \( k \). Therefore, the number of balls in the overflowing cells, \( t \), must respect the following bounds:

\[
f(\ell + 1) \leq t \leq f\ell + k
\]

Once we set the values of \( f \) and \( t \), we need to factor in the following:

1. The identities of the \( f \) (out of the possible \( m \)) cells that will overflow.
2. The identities of the \( t \) (out of the possible \( n \)) balls that will occupy the overflowing cells.
3. The number of ways to throw \( t \) balls into \( f \) cells so that all cells will overflow.
4. The number of ways to throw $n - t$ balls into $m - f$ cells so that no cell overflows. Recall that this was computed in the previous section, and was denoted by $N(n - t, m - f, \ell)$ there. Denoting the third quantity by $\bar{N}(t, f, \ell)$, we have:

$$Pr[e_{n,m,\ell} \leq k] = \frac{1}{m^n} \sum_{j=0}^{k} \binom{m}{f} \sum_{t=f(\ell+1)}^{f+g} \binom{n}{t} \bar{N}(t, f, \ell) N(n - t, m - f, \ell)$$

So we are left with the task of calculating $\bar{N}(t, f, \ell)$, the number of ways to throw $t$ balls into $f$ cells so that all cells receive more than $\ell$ balls. By the inclusion-exclusion principle,

$$\bar{N}(t, f, \ell) = \sum_{i=0}^{f} (-1)^i \binom{f}{i} \sum_{j=0}^{i} \binom{t}{j} N(j, i, \ell) (f - i)^{t-j}$$

In the sign-alternating sum, we first pick $i$ cells which will not overflow. Then, we decide how many balls these cells will hold ($0$ to $i\ell$). We choose the balls, arrange them legally in $N(j, i, \ell)$ ways, and distribute the remaining $t - j$ balls into the other $f - i$ cells arbitrarily.

**Complexity Issues** Let $l_p^k(n, m)$ denote the smallest integer $k$ such that $Pr[e_{n,m,\ell} \leq k] \geq p$. We now analyze the complexity of calculating the values of $l_p^k(n, m)$.

As discussed in Section 2, calculating $N(n, m, \ell)$ requires $\Theta(nm^2\ell)$ operations. Turning our attention to the calculation of $\bar{N}(t, f, \ell)$, we note that $f$ is confined to the range $0, \ldots, k$ and $t$ is confined to the range $0, \ldots, k\ell$. Thus, given the values of $N(n, m, \ell)$, $\bar{N}$ can be calculated in $\Theta(k^4\ell^2)$. After accounting for all values of $N$ and $\bar{N}$, we note that calculating the probability $Pr[e_{n,m,\ell} \leq k]$ requires $k^2$ operations. Thus, we require $O(n\ell \cdot k^2)$ operations in order to find $l_p^k(n, m)$, given the values of $N$ and $\bar{N}$. Overall, the complexity of calculating $l_p^k(n, m)$ is

$$O(n\ell k^2 + nm^2\ell + k^4\ell^2) = O(n^2(m^2 + k^4))$$

4 Non Optimal Merge Results

In Section 2 we computed the minimal number of results $\ell$ that the QI should fetch from each segment per query in order to ensure, with probability no less than $p \ (p < 1)$, that the $n$ merged results be the top-$n$ results for the query. Section 3 relaxed that demand to allow some of the top-$n$ results to be lost; $\ell$ was set so that with probability $p$, most of the top-$n$ results be returned. Essentially, the a-priori probabilities of losing quality results during the merge process, were studied.

Whether the QI’s policy allows some results to be lost or not, $\ell$ is set so that with probability of about $1 - p$, the merge process will fail to meet the QI’s standard of quality, losing more of the top-$n$ results than is acceptable. This section deals with such cases. Section 4.1 presents how the QI, as it is merging the $\ell \cdot m$ results returned by the segments, can estimate online how many of the top-$n$ results it may be losing. Section 4.2 discusses an additional measure of non-optimal merge results called *skipped results*, and proves a strong correlation between skipped results and lost results.

4.1 Online Estimation of the Number of Lost Results

We now show that during the merge process, the QI can assess the quality of the merged results. In particular, we show that the QI can determine the probability that at most $k$ of the top-$n$ results have been lost, replaced by less relevant results in the merge process.
Let us recall the merge process which takes place in the QI. Results from \( m \) sorted lists of size \( \ell \) are merged so as to find the top-\( n \) results for the query \( q \). As long as the merge process does not exhaust a list, that is as long as each list contributes to the best results at most \( \ell - 1 \) results, the optimality of the merge process is certain. Assume, however, that for some \( T \), the \( n - T \) best result in the merged list is the \( \ell \) (i.e., the last) result of some segment, and thus that segment’s list is exhausted. The optimality of merged results \( n - T + 1, n - T + 2, \ldots, n \) is now in doubt: result \( \ell + 1 \) of the exhausted segment, which was not retrieved by the QI and does not participate in the merge process, may match the query \( q \) better than the results which remain in the merge process. In general, result lists of several segments may be exhausted during the merge process, and we would like the QI to be able to assess the optimality of its merged results.

**Formal analysis** Let \( D_i, i = 1, \ldots, m \) denote the set of documents that are indexed on segment \( i \), and let \( \ell \) denote the number of results that each segment contributes to the merge process.

Let the ordered set \( \mathcal{B} \) denote the set of documents \( D_i \) sorted in non-increasing order according to the scores \( s_d(q), d \in D_i \) and let \( \mathcal{B}[t] \) denote the best \( t \) documents in \( \mathcal{B} \). Similarly, let \( \mathcal{R} \) denote the set of \( \ell m \) retrieved results, and let \( \mathcal{R}[t] \) denote the set of the top-\( t \) results in \( \mathcal{R} \), as determined by the merge process. Let \( R_t \) denote the rank of merged result number \( t \) with respect to the set \( \mathcal{B} \) (the absolute rank of merge result number \( t \)). Obviously, \( R_1 = 1 \) and \( R_t \geq t \). The loss at time \( t \), \( L(t) \), is defined as follows:

\[
L(t) \triangleq |\mathcal{B}[t] \setminus \mathcal{R}[t]| = |\mathcal{B}[t] \setminus \mathcal{R}|
\]

\( L(t) \) counts the number of top-\( t \) results which do not participate in (are lost by) the merge process. Clearly, \( R_t = t \iff L(t) = 0 \). This is generalized in the following Lemma:

**Lemma 1** For every \( t = 1, \ldots, n \): \( R_{t-L(t)} \leq t \) and \( R_{t-L(t)+1} > t \).

**Proof:** By definition, exactly \( t - L(t) \) results from \( \mathcal{B}[t] \) are in \( \mathcal{R} \). This means that the top \( t - L(t) \) results in \( \mathcal{R} \) are in \( \mathcal{B}[t] \), but the \( (t - L(t) + 1) \)-st result in \( \mathcal{R} \) is not in \( \mathcal{B}[t] \). The lemma follows. \( \square \)

Next, we develop a recursive formula that allows the QI to estimate, during the merge process, the probability of losing a certain number of results. For this formula we need the following notations:

Let \( t = 1, \ldots, n \); then \( \eta_t \triangleq |\{ i : D_i \cap \mathcal{R}[t] = \ell \}| \) denotes the number of exhausted segments after merging result \( t \). Similarly, define \( \mu_t \triangleq |\{ i : D_i \cap \mathcal{B}[t] \geq \ell \}| \) as the number of segments which store \( \ell \) or more of the best \( t \) documents. Note that while the values of \( \eta_t, t = 1, \ldots, n \) are known to the QI during the merge process, the corresponding values of \( \mu_t \) are not. The following Proposition links the values of \( L(t), \eta_t \) and \( \mu_t \).

**Proposition 1** For every \( t \in \{1, \ldots, n\} \), \( \mu_t = \eta_{t-L(t)} \).

**Proof:** We need to show that the document set \( D_i \) contains \( \ell \) or more results of \( \mathcal{B}[t] \) \( \iff \) segment \( i \) is exhausted before merging result \( t - L(t) + 1 \).

\( \Rightarrow \): If \( D_i \) contains at least \( \ell \) results of \( \mathcal{B}[t] \), then segment \( i \) is exhausted before the first result which is not in \( \mathcal{B}[t] \) is merged. By Lemma 1 we have that \( R_{t-L(t)+1} > t \), and hence segment \( i \) is exhausted before the \( t - L(t) + 1 \)-st result is merged, as claimed.

\( \Leftarrow \): Similarly, if by the time we merge result \( t - L(t) \) (whose rank is \( R_{t-L(t)} \)) segment \( i \) is exhausted, then the top \( \ell \) results in \( D_i \) are all ranked better than or equal to \( R_{t-L(t)} \), which by Lemma 1 is at most \( t \).

Using the above relationship, the QI can evaluate the probabilities of losing results during the merge process as follows:
Proposition 2 \( L(1) = 0, \) and for \( t = 1, \ldots, n - 1 \) and \( k = 0, \ldots, t - 1 \):

\[
Pr[L(t + 1) = k] = \frac{\eta - k + 1}{m} Pr[L(t) = k - 1] + \frac{m - \eta - k}{m} Pr[L(t) = k]
\]

Proof: For deductive convenience, assume that the query’s results are distributed to the segments from best to worst, where at step \( t \) the \( t \)-th best result is randomly inserted into one of the segments.

As noted previously, the best result is never lost and hence \( L(1) = 0 \). For larger values of \( t \), observe that either \( L(t + 1) = L(t) \) or \( L(t + 1) = L(t) + 1 \). Thus:

\[
Pr[L(t + 1) = k] = Pr[L(t) = k - 1] * Pr[L(t + 1) = k \mid L(t) = k - 1] + Pr[L(t) = k] * Pr[L(t + 1) = k \mid L(t) = k]
\]

Let \( d \) denote the \((t+1)\)-st result in \( B \). Assume that \( d \in D_j \), and let \( j_d \) denote the local rank of \( d \) in segment \( j \). Now, \( L(t + 1) = L(t) + 1 \) iff \( d \notin R \), or equivalently iff \( j_d > l \). This means that \( j, d \)’s segment, must be one of the \( \mu_t \) segments to which \( \ell \) or more of the top \( t \) results belong. Similarly, \( L(t + 1) = L(t) \iff j_d \leq l \), which means that \( j \) must be one of the \( m - \mu_t \) segments to which less than \( \ell \) of the top \( t \) results belong. Therefore,

\[
Pr[L(t + 1) = k \mid L(t) = k - 1] = \frac{\mu_t}{m} = \frac{\eta - k + 1}{m},
\]

and

\[
Pr[L(t + 1) = k \mid L(t) = k] = \frac{m - \mu_t}{m} = \frac{m - \eta - k}{m}.
\]

The proposition follows. \( \square \)

In particular, the probability that the merge process produces an optimal result set equals

\[
Pr[L(n) = 0] = \prod_{t=1}^{n-1} (1 - \frac{\eta_t}{m})
\]

The QI may set a policy that the result set should contain at most \( k \) non-optimal results with probability at least \( p_k \). In order to enforce this policy, the QI should assert that

\[
\sum_{j=0}^{k} Pr[L(n) = j] \geq p_k
\]

The QI can calculate these probabilities by updating \( Pr[L(t) = j], \ j = 0, \ldots, k \) on the fly. Whenever the merged results fail to comply with the policy, the QI may fetch more data from the segments in order to protect the quality of the merged results.

4.2 Skipped Results as a Measure of Sub-Optimality

The absolute rank of the last merged result is another indicator of the quality of the merge process. Obviously, after merging \( t \) results, \( R_t \geq t \). When \( R_t = t \), the top-\( t \) merged results are optimal; however, when \( R_t > t \), quality results have been skipped over by the merge process, allowing lesser results to occupy top-\( t \) positions in the merged results. We thus define the number of skipped results at time \( t \) by \( S(t) \triangleq R_t - t \). This number, which reflects the rank promotion of merged result \( t \) with respect to its absolute rank, also counts the number of results in the index that (a) were not merged up to time \( t \) and that (b) rank higher than \( R_t \), the rank of the result merged in time \( t \).

We begin by observing the following relationship between our two measures of non-optimal results, lost and skipped results.
Proposition 3 $L(n) \geq k$ iff $S(n - k + 1) \geq k$.

Proof: Let $L(n) \geq k$. It follows that $R_{n-k+1} > n$. Therefore,

$$S(n - k + 1) = R_{n-k+1} - (n - k + 1) > n - n + k - 1 = k - 1 \implies S(n - k + 1) \geq k$$

Conversely, let $S(n - k + 1) \geq k$. It follows that

$$R_{n-k+1} = S(n - k + 1) + (n - k + 1) \geq n + 1,$$

therefore $L(n) \geq k$.

We cannot use the above relationship to deduce probabilities of the type $Pr[S(n) = k]$ for arbitrary values of $k$. We therefore propose the following online computation for the probabilities $S(n) = j$ for $j = 0, \ldots, k$. As in the previous subsection, we again consider the query’s results to be uniformly and independently distributed to the segments from best to worst. Consider a merge process after $t \geq 1$ steps, that has just merged the result whose rank is $R_t$. Recall that $\eta_t$ is the number of exhausted segments at that time. The results ranking in place $1 + R_t, \ldots, j + R_t$ will be skipped iff each is assigned to one of the $\eta_t$ exhausted segments. We can think of each result assignment as a Bernoulli trial, which fails if the result is assigned to an exhausted segment, and succeeds otherwise. This leads to the observation that the increase in skips following merge step $t+1$, $S(t+1) - S(t)$, equals the number of failures in a series of independent Bernoulli trials, each with a success probability of $1 - \frac{\eta_t}{m}$, until the first success. In other words, $S(t+1) - S(t)$ is a geometric random variable with parameter $1 - \frac{\eta_t}{m}$. Thus, for $t > 1$ and $j = 0, \ldots, k$,

$$Pr[S(t+1) = k \mid S(t) = k-j] = \left(\frac{\eta_t}{m}\right)^j (1 - \frac{\eta_t}{m}),$$

and hence

$$Pr[S(t+1) = k] = \sum_{j=0}^{k} \left(\frac{\eta_t}{m}\right)^j (1 - \frac{\eta_t}{m}) Pr[S(t) = k-j]$$

The above derivation, along with the fact that $S(1) = 0$ with probability one, allows probabilities of the form $Pr[S(t) = j]$ for $j = 0, \ldots, k$ to be computed in $O(k^2)$ per merge step $^2$.

An alternative computation of the probability of skips, which may be more efficient when the merge process is long ($n$ is large), is the following. Let $\alpha \triangleq \eta_{n-1}$ denote the number of exhausted segments before the last (the $n$’th) merge step. Further, define

$$\tau_i = \min\{t \mid \eta_t = i+1\}, \quad i = 0, \ldots, \alpha - 1$$

i.e., merge step $\tau_i$ is the latest which was executed with exactly $i$ exhausted segment. Accordingly, define $\tau_\alpha \triangleq n$.

The first $\tau_0$ merge steps took place while no segment is exhausted, and so $R_{\tau_0} = \tau_0$ and $S_{\tau_0} = 0$. The next $\tau_1 - \tau_0$ merge steps took place while one segment was exhausted. In general, let $t_i = \tau_i - \tau_{i-1}$, $i = 1, \ldots, \alpha$; there were $t_i$ merge steps which took place while exactly $i$ segments were exhausted, and we will refer to those steps as the $i$’th phase of the merge process. We also define $X_i \triangleq S(\tau_i) - S(\tau_{i-1})$, the increase in the number of skips during phase $i$; it is easy to see that

$$S(n) = \sum_{i=1}^{\alpha} X(i) = \sum_{i=1}^{\alpha} S(\tau_i) - S(\tau_{i-1})$$

$^2$The complexity can be lowered to $O(k \log k)$ per step by applying FFT, see [4] for more details.
Phase $i$ of the merge process constitutes a sequence of Bernoulli trials, $t_i$ of which are successful, and whose failures correspond to skipped results. $X_i$ is therefore a negative binomial random variable with parameters $(t_i, 1 - \frac{i}{m})$, and

$$Pr[X_i = k] = \binom{k + t_i - 1}{k} \left(1 - \frac{i}{m}\right)^t \left(\frac{i}{m}\right)^k$$

By Equation 2 we deduce that $S(n)$ is distributed according to a convolution of a series of $\alpha$ independent (and differently distributed) negative binomial random variables:

$$Pr[S(n) = k] = Pr[\sum_{i=1}^{\alpha} X_i = k] = \sum_{j=0}^{k} Pr[X_1 = j] \cdot Pr[\sum_{i=2}^{\alpha} X_i = k - j]$$

Since every $X_j$ is a negative binomial random variable, $Pr[X_j = i]$ can be evaluated by expression (3) in $O(\log t_i + k)$ operations. A naive iterative procedure can thus calculate $Pr[S(n) = k]$ in $O(\alpha k^2) = O(mk^2)$ evaluations of such expressions, resulting in a total of $O(mk^3 \log n)$ operations.

5 Search Engines with Segmented Inverted Indices

Inverted indices, or inverted lists/files, are regarded as the most widely applied indexing technique [1, 9, 21, 18, 16, 19], and are believed to be used by the major search engines. As search engines index billions of Web pages, the size of their inverted indices is measured in terabytes.

Ribeiro-Neto and Barbosa [18] mention three hardware configurations that can handle large digital libraries: a powerful central machine, a parallel machine, or a high-speed network of machines (workstations and high end desktops). When considering the size of the indices which search engines maintain, the growth rate of the Web and the large number of queries which search engines answer each day, using a network of machines is considered to be the most cost-effective and scalable architecture [7, 18].

There are two well-studied schemes of partitioning an inverted index across several machines:

- **Global index organization.** In this scheme, the inverted index is partitioned by terms. Each machine holds posting lists for a distinct set of terms. The postings for term $t$ list all the documents that include $t$.

- **Local index organization.** In this scheme, the inverted index is partitioned by documents. Each machine is responsible for indexing a distinct set of documents, and will hold posting lists for all terms that appeared in its set of documents.

This work pertains to the local index organization scheme. Many works [3, 19, 7, 18, 21] describe essentially the same model of query execution in systems with locally-segmented indices:

- User queries arrive to a certain designated machine, which we denoted by QI. This machine was called *home site* in [21], *central broker* in [19, 18], *user interface* (or UIF) in [7] and *connection server* in [3].

---

3 A negative binomial random variable with parameters $(r, p)$ counts the number of failures in a series of i.i.d. Bernoulli trials, each with a success probability of $p$, until the $r$-th success. Negative binomials are often alternatively defined as counting the number of i.i.d. Bernoulli trials, each with success probability $p$, until the $r$-th success [6].

4 Again, this can be lowered to $O(mk^2 \log k \log n)$ by applying FFT.
• The QI issues each query to all segments, and waits for them to return their local result sets. With local index partitioning, it is usually assumed that each segment has the ability to calculate the global score of each document in its local index with respect to all queries.

• The QI merges the multiple result sets with respect to the system’s ranking scheme, and returns the merged results to the user. Since the result sets that are returned by different segments are disjoint, merging the various result sets is straightforward.

One of the parameters that impacts the complexity of the above query execution model, is the number \( \ell \) of results that each segment returns to the QI \([13]\). As \( \ell \) grows, each segment invests more resources while identifying its local top-\( \ell \) results, and the communication between the QI and the segments becomes more expensive (longer result lists must be buffered and transmitted). Therefore, optimizing \( \ell \) - which motivated the study of this paper - may lower the resources that are consumed during query executions.

6 Previous Work

The stochastic properties of the process which randomly throws \( n \) balls into \( m \) cells have been studied extensively. Two good references are \([12]\) and \([11]\). Among the properties studied was the distribution of the maximum number of balls in a cell, which we will denote by \( L(n,m) \). For example, for \( n \geq m \) (more balls than cells), \( L(n,m) = \Theta\left( \frac{\ln m}{1 + \frac{n}{\ln n}} + \frac{n}{m} \right) \) with probability \( 1 - o(1) \) \([5]\). When \( n = m \), \( L(n,m) \) behaves asymptotically as \( (1 + o(1)) \frac{\ln n}{\ln m} \) with probability \( 1 - o(1) \) \([2]\). In \([12]\), the distribution of \( L(n,m) \) is examined with regard to the behavior of the ratio \( \frac{n}{m \ln m} \) as \( n, m \to \infty \). Separate results are obtained for the three cases \( \frac{n}{m \ln m} \to 0 \), \( \frac{n}{m \ln m} \to \lambda > 0 \), and \( \frac{n}{m \ln m} \to \infty \). In \([10]\) it was shown that the distribution of \( L(n,m) \) may be approximated by the the distribution of

\[
\frac{n \cdot \max \sum_{j=1}^{m} s_j}{\sum_{j=1}^{m} s_j},
\]

where each \( s_j \) is an independent \( \chi^2 \) variable with \( \frac{2(n-1)}{m} \) degrees of freedom.

The above results are all asymptotic in nature. However, in the context of search engines, no more than 15% of users browse through more than 30 search results \([20, 8, 15, 14]\), and so \( n \) is usually a small number. The number of index segments, \( m \), is also a constant (per search engine). The calculations of Sections 2 and 3 yield the exact values of \( l_p(n,m) \) and of \( l_p^{\ell}(n,m) \), requiring polynomial-time computations.

References


