Million Query Track 2007 Overview

James Allan*, Ben Carterette*, Javed A. Aslam+,
Virgil Pavlu+, Blagovest Dachev*, and Evangelos Kanoulas+

* Center for Intelligent Information Retrieval, Department of Computer Science
University of Massachusetts Amherst, Amherst, Massachusetts

+ College of Computer and Information Science, Northeastern University
Boston, Massachusetts

[Notebook version]

The Million Query (1MQ) track ran for the first time in TREC 2007. It was designed to serve two purposes. First, it was an exploration of ad-hoc retrieval on a large collection of documents. Second, it investigated questions of system evaluation, particularly whether it is better to evaluate using many shallow judgments or fewer thorough judgments.

Participants in this track were assigned two tasks: (1) run 10,000 queries against a 426Gb collection of documents at least once and (2) judge documents for relevance with respect to some number of queries.

Section 1 describes how the corpus and queries were selected, details the submission formats, and provides a brief description of all submitted runs. Section 2 provides an overview of the judging process, including a sketch of how it alternated between two methods for selecting the small set of documents to be judged. Sections 3 and 4 provide details of those two selection methods, developed at UMass and NEU, respectively. The sections also provide some analysis of the results.

In Section 6 we present some statistics about the judging process, such as the total number of queries judged, how many by each approach, and so on. We present some additional results and analysis of the overall track in Sections 7 and 8.

1 Phase I: Running Queries

The first phase of the track required that participating sites submit their retrieval runs.

1.1 Corpus

The 1MQ track used the so-called "terabyte" or "GOV2" collection of documents. This corpus is a collection of Web data crawled from Web sites in the .gov domain in early 2004. The collection is believed to include a large proportion of the .gov pages that were crawlable at that time, including HTML and text, plus the extracted text of PDF, Word, and PostScript files. Any document longer than 256Kb was truncated to that size at the time the collection was built. Binary files are not included as part of the collection, though were captured separately for use in judging.

The GOV2 collection includes 25 million documents in 426 gigabytes. The collection was made available by the University of Glasgow, distributed on a hard disk that was shipped to participants for an amount intended to cover the cost of preparing and shipping the data.

1.2 Queries

Topics for this task were drawn from a large collection of queries that were collected by a large Internet search engine. Each of the chosen queries is likely to have at least one relevant document in the GOV2 collection

because logs showed a clickthrough on one page captured by GOV2. Obviously there is no guarantee that the clicked page is relevant, but it increases the chance of the query being appropriate for the collection.

These topics are short, title-length (in TREC parlance) queries. In the judging phase, they were developed into full-blown TREC topics.

Ten thousand (10,000) queries were selected for the official run. The 10,000 queries included 150 queries that were judged in the context of the Terabyte Track from earlier years (though one of these had no relevant documents and was therefore excluded).

No quality control was imposed on the 10,000 selected queries. The hope was that most of them would be good quality queries, but it was recognized that some were likely to be partially or entirely non-English, to contain spelling errors, or even to be incomprehensible to anyone other than the person who originally created them.

The queries were distributed in a text file where each line has the format "N:query word or words". Here, N is the query number, is followed by a colon, and immediately followed by the query itself. For example, the line (from a training query) "32:barack obama internships" means that query number 32 is the 3-word query "barack obama internships". All queries were provided in lowercase and with no punctuation (it is not clear whether that formatting is a result of processing or because people use lowercase and do not use punctuation).

1.3 Submissions

Sites were permitted to provide up to five runs. Every submitted run was included in the judging pool and all were treated equally.

A run consisted of up to the top 1,000 documents for each of the 10,000 queries. The submission format was a standard TREC format of exactly six columns per line with at least one space between the columns. For example:

```
100 Q0 ZF08-175-870 1 9876 mysys1
100 Q0 ZF08-306-044 2 9875 mysys2
```

where:

1. The first column is the topic number.
2. The second column is unused but must always be the string "Q0" (letter Q, number zero).
3. The third column is the official document number of the retrieved document, found in the <DOCNO> field of the document.
4. The fourth column is the rank of that document for that query.
5. The fifth column is the score this system generated to rank this document.
6. The six column was a "run tag," a unique identifier for each group and run.

If a site would normally have returned no documents for a query, it instead returned the single document "GX000-00-0000000" at rank one. Doing so maintained consistent evaluation results (averages over the same number of queries) and did not break any evaluation tools being used.

1.4 Submitted runs

The following is a brief summary of some of the submitted runs. The summaries were provided by the sites themselves and are listed in alphabetical order. (When no full summary is available, the brief summary information from the submissions has been used.)

ARSC/University of Alaska Fairbanks The ARSC multisearch system is a heterogeneous distributed information retrieval simulation and demonstration implementation. The purpose of the simulation is to illustrate performance issues in Grid Information Retrieval applications by partitioning the GOV2 collection into a large number of hosts and searching each host independently of the others. Previous

TREC Terabyte Track experiments using the ARSC multisearch system have focused on the IR performance of multisearch result-set merging and the efficiency gains from truncating result-sets from a large collection of hosts before merging.

The primary task of the ARSC multisearch system in the 2007 TREC Million Query experiment is to estimate the number of hosts or subcollections of GOV2 that can be used to process 10,000 queries within the TREC Million Query Track time constraints. The secondary and ongoing task is to construct an effective strategy for picking a subsets of the GOV2 collections to search at query-time. The host-selection strategy used for this experiment was to restrict searches to hosts that returned the most relevant documents in previous TREC Terabyte Tracks.

Exegy Exegy's submission for the TREC 2007 million query track consisted of results obtained by running the queries against the raw data, i.e., the data was not indexed. The hardware-accelerated streaming engine used to perform the search is the Exegy Text Miner (XTM), developed at Exegy, inc. The search engine's architecture is novel: XTM is a hybrid system (heterogeneous compute platform) employing general purpose processors (GPPs) and field programmable gate arrays (FPGAs) in a hardware-software co-design architecture to perform the search. The GPPs are responsible for inputting the data to the FPGAs and reading and post-processing the search results that the FPGAs output. The FPGAs perform the actual search and due to the high degree of parallelism available (including pipelining) are able to do so much more efficiently than the GPP.

For the million query track the results for a particular query were obtained by searching for the exact query string within the corpus. This brute force approach, although naïve, returned relevant results for most of the queries. The mean-average precision for the results was 0.3106 and 0.0529 using the UMass and the NEU approaches, respectively. More importantly, XTM completed the search for the entire set of the 10,000 queries on the unindexed data in less than two and a half hours.

Heilongjiang Institute of Technology, China Used Lemur.

IBM Haifa This year, the experiments of IBM Haifa were focused on the scoring function of Lucene, an Apache open-source search engine. The main goal was to bring Lucene's ranking function to the same level as the state-of-the-art ranking formulas like those traditionally used by TREC participants. Lucene's scoring function was modified to include better document length normalization, and a better term-weight setting following to the SMART model.

Lucene then compared to Juru, the home-brewed search engine used by the group in previous TREC conferences. In order to examine the ranking function alone, both Lucene and Juru used the same HTML parser, the same anchor text, and the same query parsing process including stop-word removal, synonym expansion, and phrase expansion. Based on the 149 topics of the Terabyte tracks, the results of modified Lucene significantly outperform the original Lucene and are comparable to Juru's results.

In addition, a shallow query log analysis was conducted over the 10K query log. Based on the query log, a specific stop-list and a synonym-table were constructed to be used by both search engines.

Northeastern University We used several standard Lemur built in systems (tfidf_bm25, tfidf_log, kl_abs,kl_dir,inquery,cos, okapi) and combined their output (metasearch) using the hedge algorithm.

RMIT Zettair Dirichlet smoothed language model run.

SabIR Standard smart ltu.Lnu run.

University of Amsterdam The University of Amsterdam, in collaboration with the University of Twente, participated with the main aim to compare results of the earlier Terabyte tracks to the Million Query track. Specifically, what is the impact of shallow pooling methods on the (apparent) effectiveness of retrieval techniques? And what is the impact of substantially larger numbers of topics? We submitted a number of runs using different document representations (such as full-text, title-fields, or incoming anchor-texts) to increase pool diversity. The initial results show broad agreement in system rankings over various measures on topic sets judged at both Terabyte and Million Query tracks, with runs using the full-text index giving superior results on all measures. There are some noteworthy upsets: measures using the Million Query judged topics show stronger correlation with precision at early ranks.

University of Massachusetts Amherst The base UMass Amherst submissions were a simple query likelihood model and the dependence model approach fielded during the terabyte track last year. We also tried some simple automatic spelling correction on top of each baseline to deal with errors of that kind. All runs were done using the Indri retrieval system.

University of Melbourne Four types of runs were submitted:

1. A topic-only run using a similarity metric based on a language model with Dirichlet smoothing as describe by Zhai and Lafferty (2004).
2. Submit query to public web search engine, retrieve snippet information for top 5 documents, add unique terms from snippets to query, run expanded query using same similarity metric just described.
3. A standard impact-based ranking.
4. A merging of the language modeling and the impact runs.

2 Phase I: Relevance judgments and judging

After all runs were submitted, a subset of the topics were judged. The goal was to provide a small number of judgments for a large number of topics. For TREC 2007, over 1700 queries were judged, a large increase over the more typical 50 queries judged by other tracks in the past.

2.1 Judging overview

Judging was done by assessors at NIST and by participants in the track. Non-participants were welcome (encouraged!) to provide judgments, too, though very few such judgments occurred. Some of the judgments came from an Information Retrieval class project, and some were provided by hired assessors at UMass. The bulk of judgments, however, came from the NIST assessors.

The process looked roughly like this from the perspective of someone judging:

1. The assessment system presented 10 queries randomly selected from the evaluation set of 10,000 queries.
2. The assessor selected one of those ten queries to judge. The others were returned to the pool.
3. The assessor provided the description and narrative parts of the query, creating a full TREC topic. This information was used by the assessor to keep focus on what is relevant.
4. The system presented a GOV2 document (Web page) and asked whether it was relevant to the query. Judgments were on a three-way scale to mimic the Terabyte Track from years past: highly relevant, relevant, or not relevant. Consistent with past practice, the distinction between the first two was up to the assessor.
5. The assessor was required to continue judging until 40 documents has been judged. An assessor could optionally continue beyond the 40, but few did.

The system for carrying out those judgments was built at UMass on top of the Drupal content management platform[1]. The same system was used as the starting point for relevance judgments in the Enterprise track.

2.2 Selection of documents for judging

Two approaches to selecting documents were used:

Expected AP method. In this method, documents are selected by how much they inform us about the difference in mean average precision given all the judgments that were made up to that point [10]. Because average precision is quadratic in relevance judgments, the amount each relevant document

[1] http://drupal.org

contributes is a function of the total number of judgments made and the ranks they appear at. Non-relevant documents also contribute to our knowledge: if a document is nonrelevant, it tells us that certain terms cannot contribute anything to average precision. We quantify how much a document will contribute if it turns out to be relevant or nonrelevant, then select the one that we expect to contribute the most. This method is further described below in Section 3.

Statistical evaluation method. This method draws and judges a specific random sample of documents from the given ranked lists and produces unbiased, low-variance estimates of average precision, R-precision, and precision at standard cutoffs from these judged documents [1]. Additional (non-random) judged documents may also be included in the estimation process, further improving the quality of the estimates. This method is further described below in Section 4.

For each query, one of the following happened:

1. The pages to be judged for the query were selected by the "expected AP method." A minimum of 40 documents were judged, though the assessor was allowed to continue beyond 40 if so motivated.

2. The pages to be judged for the query were selected by the "statistical evaluation method." A minimum of 40 documents were judged, though the assessor was allowed continue beyond 40 if so motivated.

3. The pages to be judged were selected by alternating between the two methods until each has selected 20 pages. If a page was selected by more than one method, it was presented for judgment only once. The process continues until at least 40 pages have been judged (typically 20 per method), though the assessor was allowed continue beyond 40 if so motivated. (See Section 5.)

The assignments were made such that option (3) was selected half the time and the other two options each occurred 1/4 of the time. When completed, roughly half of the queries therefore had parallel judgments of 20 or more pages by each method, and the other half had 40 or more judgments by a single method.

In addition, a small pool of 50 queries were randomly selected for multiple judging. With a small random chance, the assessor's ten queries were drawn from that pool rather than the full pool. Whereas in the full pool no query was considered by more than one person, in the multiple judging pool, a query could be considered by any or even all assessors—though no assessor was shown the same query more than once.

3 UMass Method

The UMass algorithm is a greedy anytime algorithm. It iteratively orders documents according to how much information they provide about a difference in average precision, presents the top document to be judged, and, based on the judgment, re-weights and re-orders the documents.

Algorithm 3.1 shows the high-level pseudo-code for the algorithm, which we call MTC for *minimal test collection*.

Algorithm 3.1 MTC($\mathcal{S}, \mathcal{Q}$)

Require: a set of ranked lists $\mathcal{S}$, a set of qrels $\mathcal{Q}$ (possibly empty)

```
q = GET-QRELS(Q)
w = INIT-WEIGHTS(S, q)
loop
    i* = arg max_i w
    request judgment for document i*
    receive judgment j_{i*} for document i*
    w = UPDATE-WEIGHTS(i*, S)
    q_{i*} = j_{i*}
```

Here $\mathbf{q}$ is a vector of relevance judgments read in from a *qrels* file if one exists (for example if an assessor is resuming judging a topic that he had previously stopped). GET-QRELS simply translates (document, judgment) pairs into vector indexes such that $q_i = 1$ if the document has been judged relevant and 0

otherwise; if an assessor is just starting a topic, q_i will be 0 for all i. $\mathbf{w}$ is a vector of document weights (see below). We assume that there's a global ordering of documents, so that the relevance of document i can be found at index i in $\mathbf{q}$, and its weight at the same index in $\mathbf{w}$.

The INIT-WEIGHTS, SET-WEIGHTS, and UPDATE-WEIGHTS functions are where the real work happens. The pseudo-code below is rather complicated, so first some notational conventions: We shall use $i, j = 1 \cdots n$ to enumerate n documents and $s = 1 \cdots m$ to enumerate m systems. Capital bold letters are matrices. Column and row vectors for a matrix $\mathbf{M}$ are denoted $\mathbf{M}_{\cdot i}$ (for the ith column vector) or $\mathbf{M}_{i\cdot}$ (for the ith row vector). Matrix cells are referred to with nonbold subscripted letters, e.g. M_{ij}. Lowercase bold letters are vectors, and lowercase nonbold letters are scalars. Superscripts are never exponents, always some type of index.

Algorithm 3.2 INIT-WEIGHTS($\mathcal{S}, \mathbf{q}$)

Require: ranked lists $\mathcal{S}$, a vector of judgments $\mathbf{q}$

1: $\mathbf{V}^R = \mathbf{V}^N = [0]_{n \times m}$
2: **for all** $s \in \mathcal{S}$ **do**
3: $\quad \mathbf{C} = [0]_{n \times n}$
4: $\quad$ **for all** pairs of documents (i, j) **do**
5: $\quad\quad C_{ij} = 1/\max\{r_s(i), r_s(j)\}$
6: $\quad \mathbf{V}^R_{\cdot s} = \mathbf{C}\mathbf{q} + diag(\mathbf{C})$
7: $\quad \mathbf{V}^N_{\cdot s} = \mathbf{C}(1 - \mathbf{q})$
8: **return** SET-WEIGHTS()

Algorithm 3.3 SET-WEIGHTS()

Require: access to global weight matrices $\mathbf{V}^R, \mathbf{V}^N$

1: $\mathbf{w} = [0]_n$
2: **for all** unjudged documents i **do**
3: $\quad w^R_i = \max \mathbf{V}^R_{i\cdot} - \min \mathbf{V}^R_{i\cdot}$
4: $\quad w^N_i = \max \mathbf{V}^N_{i\cdot} - \min \mathbf{V}^N_{i\cdot}$
5: $\quad w_i = \max\{w^R_i, w^N_i\}$
6: **return** $\mathbf{w}$

Algorithm 3.2 initializes the weight vector. At line 1 we create two "global" weight matrices in which each element V_{is} is the effect a judgment will have on the average precision of system s (see below for more detail). We iterate over systems (line 2), for each run creating a coefficient matrix $\mathbf{C}$ (lines 3–5). Each pair of documents has an associated coefficient $1/\max\{r_s(i), r_s(j)\}$, where $r_s(i)$ is the rank of document i in system s (infinity if document i is unranked). In lines 6 and 7, we multiply the coefficient matrix by the qrels vector and assign the resulting vector to the corresponding system column of the weight matrix. At the end of this loop, the matrices $\mathbf{V}^R, \mathbf{V}^N$ contain the individual system weights for every document. Each column s contains the weights for system s and each row i the weights for document i.

The global weight of a document is the maximum difference between pairs of system weights. Global weights are set with the SET-WEIGHTS function, shown in Algorithm 3.3. For each row in the weight matrices, it finds the maximum and minimum weights in any system. The difference between these is the maximum pairwise difference. Then the maximum of w^R_i and w^N_i is the final weight of the document.

After each judgment, UPDATE-WEIGHTS (Algorithm 3.4) is called to update the global weight matrices and recomputes the document weights. $\mathbf{C}'$ is constructed by pulling the i^*th column from each of the m coefficient matrices $\mathbf{C}$ defined in set-weights. We construct it from scratch rather than keep all m $\mathbf{C}$ matrices in memory. Global weight matrices are updated simply by adding or subtracting $\mathbf{C}'$ depending on the judgment to i^*.

3.1 Running Time

MTC loops until the assessor quits or all documents have been judged. Within the loop, finding the maximum-weight document (line 4) is in $\mathcal{O}(n)$. UPDATE-WEIGHTS loops over systems and documents for a

Algorithm 3.4 UPDATE-WEIGHTS($i^*, \mathcal{S}$)

Require: the index of the most recent judgment i^*, a set of ranked lists $\mathcal{S}$
Require: access to global weight matrices $\mathbf{V}^R, \mathbf{V}^N$
1: $\mathbf{C}' = [0]_{n \times m}$
2: **for** $s \in \mathcal{S}$ **do**
3: **for all** documents i, $C'_{is} = 1/\max\{r_s(i^*), r_s(i)\}$
4: **if** i^* is relevant **then**
5: $\mathbf{V}^R = \mathbf{V}^R + \mathbf{C}'$
6: **else**
7: $\mathbf{V}^N = \mathbf{V}^N - \mathbf{C}'$
8: **return** SET-WEIGHTS()

runtime in $\mathcal{O}(m \cdot n)$. SET-WEIGHTS is also in $\mathcal{O}(n \cdot m)$: each max or min is over m elements, and four of them happen n times. Therefore the total runtime for each iteration is in $\mathcal{O}(m \cdot n)$.

INIT-WEIGHTS is in $\mathcal{O}(m \cdot n^2)$: we loop over m systems, each time performing $\mathcal{O}(n^2)$ operations to construct $\mathbf{C}$ and perform matrix-vector multiplication. Since MTC can iterate up to n times, the total runtime is in $\mathcal{O}(m \cdot n^2)$.

In practice, the algorithm was fast enough that assessors experienced no noticeable delay between submitting a judgment and receiving the next document, even though an entire $\mathcal{O}(m \cdot n)$ iteration takes place in between. INIT-WEIGHTS was slow enough to be noticed, but it ran only once, in the background while assessors defined a topic description and narrative.

3.2 Explanation

The pseudo-code is rather opaque, and it may not be immediately clear how it implements the algorithm described in our previous work. Here is the explanation.

In previous work we showed $AP_s \propto \sum_i \sum_j A_{ij} X_i X_j$, where X_i is a binary indicator of the relevance of document i and $A_{ij} = 1/\max\{r_s(i), r_s(j)\}$. See Section 3.3.1 for more details.

Define a lower bound for AP_s in which every unjudged document is assumed to be nonrelevant. An upper bound is similarly defined by assuming every unjudged document relevant. Denote the bounds $\lfloor AP_s \rfloor$ and $\lceil AP_s \rceil$ respectively.

Consider document i, ranked at $r_s(i)$ by system s. If we judge it relevant, $\lfloor AP_s \rfloor$ will increase by $\sum_{j|x_j=1} a_{ij} x_j$. If we judge it nonrelevant, $\lceil AP_s \rceil$ will decrease by $\sum_{j|x_j \neq 0} a_{ij} x_j$. These are matrix elements V^R_{is} and V^N_{is} respectively, computed at steps 4–7 in INIT-WEIGHTS and steps 2–7 in UPDATE-WEIGHTS.

Now suppose we have two systems s_1 and s_2. We want to judge the document that's going to have the greatest effect on $\Delta AP = AP_{s_1} - AP_{s_2}$. We can bound ΔAP as we did AP above, but the bounds are much hard to compute exactly. It turns out that that does not matter: it can be proven that the judgment that reduces the upper bound of ΔAP the most is a nonrelevant judgment to the document that maximizes $V^N_{is_1} - V^N_{is_2}$, and the judgment that increases the lower bound the most is a relevant judgment to the document that maximizes $V^R_{is_1} - V^R_{is_2}$. Since we of course do not know the judgment in advance, the final weight of document i is the maximum of these two quantities.

When we have more than two systems, we simply calculate the weight for each pair and take the maximum over all pairs as the document weight. Since the maximum over all pairs is simply the maximum weight for any system minus the minimum weight for any system, this can be calculated in linear time, as steps 3–5 of set-weights show.

3.3 UMass Evaluation

The evaluation tool `mtc-eval` takes as input one or more retrieval systems. It calculates $\mathcal{E}MAP$ (Eq. 1 below) for each system; these are used to rank the systems. Additionally, it computes $E[\Delta AP]$, $Var[\Delta AP]$, and $P(\Delta AP < 0)$ (Eqs. 2, 3, 4 respectively) for each topic and each pair of systems, and $\mathcal{E}\Delta MAP$, $\mathcal{V}\Delta MAP$, and $P(\Delta MAP < 0)$ (Eqs. 5, 6, 7 respectively) for each pair of systems. More details are provided below.

3.3.1 Expected Mean Average Precision

As we showed in Carterette et al., average precision can be written as a quadratic equation over Bernoulli trials X_i for the relevance of document i:

$$AP_s = \frac{1}{|R|} \sum_{i=1}^{n} \sum_{j \geq i} A_{ij} X_i X_j$$

where $A_{ij} = 1/\max\{r_s(i), r_s(j)\}$.

Let $p_i = p(X_i = 1)$. The expectation of AP_s is:

$$E[AP_s] \approx \frac{1}{\sum p_i} \sum_{i}^{n} \left(A_{ii} p_i + \sum_{j>i} A_{ij} p_i p_j \right)$$

We can likewise define the expected value of MAP, $\mathcal{E}MAP$, by summing over many topics:

$$\mathcal{E}MAP_s = \sum_{t \in \mathcal{T}} E[AP_{st}] \tag{1}$$

Systems submitted to the track were ranked by $\mathcal{E}MAP$. Probabilities p_i can be estimated in several different ways; Section 3 describes the method we used in detail.

3.3.2 ΔMAP and Confidence

In our previous work we have been more interested in the difference in MAP between two systems rather than the MAPs themselves. In this section we describe ΔMAP and the idea of confidence that an observed difference between systems is "real".

As in Section 3.2, suppose we have two retrieval systems s_1 and s_2. Define $\Delta AP = AP_{s_1} - AP_{s_2}$. We can write ΔAP in closed form as:

$$\Delta AP = \sum_{i=1}^{n} \sum_{j \geq i} C_{ij} X_i X_j$$

where $C_{ij} = 1/\max\{r_{s_1}(i), r_{s_1}(j)\} - 1/\max\{r_{s_2}(i), r_{s_2}(j)\}$.

ΔAP has a distribution over all possible assignments of relevance to the unjudged documents. Some assignments will result in $\Delta AP < 0$, some in $\Delta AP > 0$; if we believe that $\Delta AP < 0$ but there are many possible sets of judgments that could result in $\Delta AP > 0$, then we should say that we have low confidence in our belief.

As it turns out, ΔAP converges to a normal distribution. This makes it very easy to determine confidence: we simply calculate the expectation and variance of ΔAP and plug them into the normal cumulative density function provided by any statistics software package.

The expectation and variance of ΔAP are:

$$E[\Delta AP] = \frac{1}{\sum p_i} \sum_{i} \left(C_{ii} p_i + \sum_{j>i} C_{ij} p_i p_j \right) \tag{2}$$

$$\begin{aligned} Var[\Delta AP] = \frac{1}{\left(\sum p_i\right)^2} \Big(& \sum_i C_{ii}^2 p_i q_i + \sum_{j>i} C_{ij}^2 p_i p_j (1 - p_i p_j) \\ & + \sum_{i \neq j} 2 C_{ii} C_{ij} p_i p_j q_i + \sum_{k>j \neq i} 2 C_{ij} C_{ik} p_i p_j p_k q_i \Big) \end{aligned} \tag{3}$$

Confidence in a difference in average precision is then defined as

$$\text{confidence} = P(\Delta AP < 0) = \Phi \left(\frac{-E[\Delta AP]}{\sqrt{Var[\Delta AP]}} \right) \tag{4}$$

where Φ is the normal cumulative density function.

This can be very easily extended to determining our confidence in a difference in MAP. The expectation and variance of ΔMAP are:

$$\mathcal{E}\Delta MAP = \frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}} E[\Delta AP_t] \tag{5}$$

$$\mathcal{V}\Delta MAP = \frac{1}{|\mathcal{T}|^2}\sum_{t\in\mathcal{T}} Var[\Delta AP_t] \tag{6}$$

and

$$\text{confidence} = P(\Delta MAP < 0) = \Phi\left(\frac{-\mathcal{E}\Delta MAP}{\sqrt{\mathcal{V}\Delta MAP}}\right) \tag{7}$$

3.3.3 Estimating Relevance

The formulas above require probabilities of relevance for unjudged documents. We used the "expert aggregation" model described in [9]. We will not present details here, but the goal is to estimate the relevance of unjudged documents based on the performance of systems over the judged documents. The model takes into account:

1. the relative frequency of relevant and nonrelevant documents for a topic;
2. the ability of a system to retrieve relevant documents;
3. the ability of a system to rank relevant documents highly;
4. the ability of a system to not retrieve nonrelevant documents;
5. variance over different systems using similar algorithms to rank.

Fitting the model is a three-step process: first, ranks are mapped to decreasing probabilities based on the number of judged relevant and judged nonrelevant documents identified for each topic. Second, these probabilities are calibrated to each system's ability to retrieve relevant documents at each rank. Finally, the systems' calibrated probabilities and the available judgments are used to train a logistic regression classifier for relevance. The model predicts probabilities of relevance for all unjudged documents.

4 NEU Evaluation Method

In this section, we describe the statistical sampling evaluation methodology, *statAP*, developed at Northeastern University and employed in the Million Query track. We begin with a simple example in order to provide intuition for the sampling strategy ultimately employed, and we then proceed to describe the specific application of this intuition to the general problem of retrieval evaluation.

4.1 Sampling Theory and Intuition

As a simple example, suppose that we are given a ranked list of documents $(d_1, d_2, \ldots)$, and we are interested in determining the precision-at-cutoff 1000, i.e., the fraction of the top 1000 documents that are relevant. Let $PC(1000)$ denote this value. One obvious solution is to examine each of the top 1000 documents and return the number of relevant documents seen divided by 1000. Such a solution requires 1000 relevance judgments and returns the *exact* value of $PC(1000)$ with *perfect certainty*. This is analogous to forecasting an election by polling each and every registered voter and asking how they intend to vote: In principle, one would determine, with certainty, the exact fraction of voters who would vote for a given candidate on that day. In practice, the cost associated with such "complete surveys" is prohibitively expensive. In election forecasting, market analysis, quality control, and a host of other problem domains, *random sampling* techniques are used instead [13].

In random sampling, one trades-off *exactitude* and *certainty* for *efficiency*. Returning to our $PC(1000)$ example, we could instead *estimate* $PC(1000)$ with some *confidence* by sampling in the obvious manner: Draw m documents uniformly at random from among the top 1000, judge those documents, and return the number of relevant documents seen divided by m — this is analogous to a random poll of registered voters in election forecasting. In statistical parlance, we have a *sample space* of documents indexed by $k \in \{1, \ldots, 1000\}$, we have a *sampling distribution* over those documents $p_k = 1/1000$ for all $1 \leq k \leq 1000$, and we have a *random variable* X corresponding to the relevance of documents,

$$x_k = rel(k) = \begin{cases} 0 & \text{if } d_k \text{ is non-relevant} \\ 1 & \text{if } d_k \text{ is relevant.} \end{cases}$$

One can easily verify that the *expected value* of a single random draw is $PC(1000)$

$$E[X] = \sum_{k=1}^{1000} p_k \cdot x_k = \frac{1}{1000} \sum_{k=1}^{1000} rel(k) = PC(1000),$$

and the Law of Large Numbers and the Central Limit Theorem dictate that the *average* of a set S of m such random draws

$$\widehat{PC}(1000) = \frac{1}{m} \sum_{k \in S} X_k = \frac{1}{m} \sum_{k \in S} rel(k)$$

will converge to its expectation, $PC(1000)$, quickly [11] — this is the essence of random sampling.

Random sampling gives rise to a number of natural questions: (1) How should the random sample be drawn? In *sampling with replacement*, each item is drawn independently and at random according to the distribution given (uniform in our example), and repetitions may occur; in *sampling without replacement*, a random subset of the items is drawn, and repetitions will not occur. While the former is much easier to analyze mathematically, the latter is often used in practice since one would not call the same registered voter twice (or ask an assessor to judge the same document twice) in a given survey. (2) How should the sampling distribution be formed? While $PC(1000)$ seems to dictate a uniform sampling distribution, we shall see that non-uniform sampling gives rise to much more efficient and accurate estimates. (3) How can one quantify the accuracy and confidence in a statistical estimate? As more samples are drawn, one expects the accuracy of the estimate to increase, but by how much and with what confidence? In the paragraphs that follow, we address each of these questions, in reverse order.

While statistical estimates are generally designed to be correct in expectation, they may be high or low in practice (especially for small sample sizes) due to the nature of random sampling. The variability of an estimate is measured by its *variance*, and by the Central Limit Theorem, one can ascribe 95% confidence intervals to a sampling estimate given its variance. Returning to our $PC(1000)$ example, suppose that (unknown to us) the actual $PC(1000)$ was 0.25; then one can show that the variance in our random variable X is 0.1875 and that the variance in our sampling estimate is $0.1875/m$, where m is the sample size. Note that the variance decreases as the sample size increases, as expected. Given this variance, one can derive 95% confidence intervals [11], i.e., an error range within which we are 95% confident that our estimate will lie.[2] For example, given a sample of size 50, our 95% confidence interval is $+/-0.12$, while for a sample of size 500, our 95% confidence interval is $+/-0.038$. This latter result states that with a sample of size 500, our estimate is likely to lie in the range $[0.212, 0.288]$. In order to increase the accuracy of our estimates, we must decrease the size of the confidence interval. In order to decrease the size of the confidence interval, we must decrease the variance in our estimate, $0.1875/m$. This can be accomplished by either (1) decreasing the variance of the underlying random variable X (the 0.1875 factor) or (2) increasing the sample size m. Since increasing m increases our judgment effort, we shall focus on decreasing the variance of our random variable instead.

While our $PC(1000)$ example seems to inherently dictate a uniform sampling distribution, one can reduce the variance of the underlying random variable X, and hence the sampling estimate, by employing *non-uniform* sampling. A maxim of sampling theory is that accurate estimates are obtained when one samples with *probability proportional to size* (PPS) [13]. Consider our election forecasting analogy: Suppose that

[2] For estimates obtained by averaging a random sample, the 95% confidence interval is roughly $+/-1.965$ *standard deviations*, where the standard deviation is the square root of the variance, i.e., $\sqrt{0.1875/m}$ in our example.

our hypothetical candidate is known to have strong support in rural areas, weaker support in the suburbs, and almost no support in major cities. Then to obtain an accurate estimate of the vote total (or fraction of total votes) this candidate is likely to obtain, it makes sense to spend your (sampling) effort "where the votes are." In other words, one should spend the greatest effort in rural areas to get very accurate counts there, somewhat less effort in the suburbs, and little effort in major cites where very few people are likely to vote for the candidate in question. However, one must now compensate for the fact that the sampling distribution is non-uniform — if one were to simply return the fraction of polled voters who intend to vote for our hypothetical candidate when the sample is highly skewed toward the candidate's areas of strength, then one would erroneously conclude that the candidate would win in a landslide. To compensate for non-uniform sampling, one must *under-count* where one *over-samples* and *over-count* where one *under-samples*.

Returning to our $PC(1000)$ example, employing a PPS strategy would dictate sampling "where the relevant documents are." Analogous to the election forecasting problem, we do have a prior belief about where the relevant documents are likely to reside — in the context of ranked retrieval, relevant documents are generally more likely to appear toward the top of the list. We can make use of this fact to reduce our sampling estimate's variance, so long as our assumption holds. Consider the non-uniform sampling distribution shown in Figure 1 where

$$p_k = \begin{cases} 1.5/1000 & 1 \leq k \leq 500 \\ 0.5/1000 & 501 \leq k \leq 1000. \end{cases}$$

Here we have *increased* our probability of sampling the top half (where more relevant documents are likely to reside) and *decreased* our probability of sampling the bottom half (where fewer relevant documents are likely to reside).

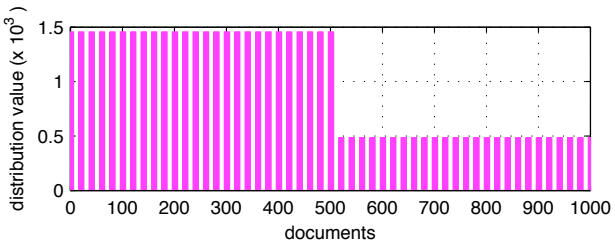

Figure 1: Non-uniform sampling distribution.

In order to obtain the correct estimate, we must now "under-count" where we "over-sample" and "over-count" where we "under-sample." This is accomplished by modifying our random variable X as follows:

$$x_k = \begin{cases} rel(k)/1.5 & 1 \leq k \leq 500 \\ rel(k)/0.5 & 501 \leq k \leq 1000. \end{cases}$$

Note that we over/under-count by precisely the factor that we under/over-sample; this ensures that the expectation is correct:

$$\begin{aligned} E[X] &= \sum_{k=1}^{1000} p_k \cdot x_k = \sum_{k=1}^{500} \frac{1.5}{1000} \cdot \frac{rel(k)}{1.5} + \sum_{k=1}^{500} \frac{0.5}{1000} \cdot \frac{rel(k)}{0.5} \\ &= \frac{1}{1000} \sum_{k=1}^{1000} rel(k) = PC(1000). \end{aligned}$$

For a given sample S of size m, our estimator is then a weighted average

$$\begin{aligned} \widehat{PC}(1000) &= \frac{1}{m} \sum_{k \in S} X_k \\ &= \frac{1}{m} \left(\sum_{k \in S \,:\, k \leq 500} \frac{rel(k)}{1.5} + \sum_{k \in S \,:\, k > 500} \frac{rel(k)}{0.5} \right) \end{aligned}$$

where we over/under-count appropriately.

Note that our expectation and estimator are correct, *independent of whether our assumption about the location of the relevant documents actually holds!* However, if our assumption holds, then the variance of our

random variable (and sampling estimate) will be reduced (and vice versa). Suppose that all of the relevant documents were located where we over-sample. Our expectation would be correct, and one can show that the variance of our random variable is reduced from 0.1875 to 0.1042 — we have sampled where the relevant documents are and obtained a more accurate count as a result. This reduction in variance yields a reduction in the 95% confidence interval for a sample of size 500 from $+/-0.038$ to $+/-0.028$, a 26% improvement. Conversely, if the relevant documents were located in the bottom half, the confidence interval would increase.

One could extend this idea to three (or more) strata, as in Figure 2. For each document k, let α_k be the factor by which it is over/under-sampled with respect to the uniform distribution; for example, in Figure 1, α_k is 1.5 or 0.5 for the appropriate ranges of k, while in Figure 2, α_k is 1.5, 1, or 0.5 for appropriate ranges of k. For a sample S of size m drawn according to the distribution in question, the sampling estimator would be

$$\widehat{PC}(1000) = \frac{1}{m} \sum_{k \in S} \frac{rel(k)}{\alpha_k}.$$

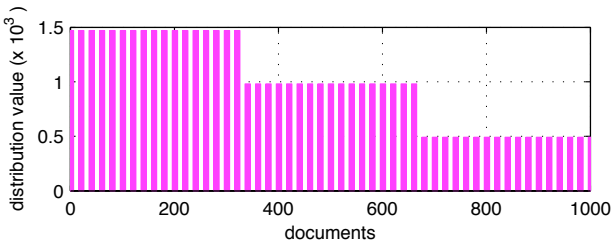

Figure 2: Non-uniform distrib. with three strata.

In summary, one can sample with respect to any distribution, and so long as one over/under-counts appropriately, the estimator will be correct. Furthermore, if the sampling distribution places higher weight on the items of interest (e.g., relevant documents), then the variance of the estimator will be reduced, yielding higher accuracy. Finally, we note that sampling is often performed *without replacement* [13]. In this setting, the estimator changes somewhat, though the principles remain the same: sample where you think the relevant documents are in order to reduce variance and increase accuracy. The α_k factors are replaced by *inclusion probabilities* π_k, and the estimator must be normalized by the size of the sample space:

$$\widehat{PC}(1000) = \frac{1}{1000} \sum_{k \in S} \frac{rel(k)}{\pi_k}.$$

Modularity. The evaluation and sampling modules are completely independent: the sampling module produces the sample in a specific format but does not impose or assume a particular evaluation being used; conversely, the evaluation module uses the given sample, with no knowledge of or assumptions about the sampling strategy strategy empolyed (a strong improvement over method presented in [5]). In fact, the sampling technique used is known to work with many other estimators, while the estimator used is known to work with other sampling strategies [8]. This flexibility is particularly important if one has reason to believe that a different sampling strategy might work better for a given evaluation.

4.2 The sample

The sample is the set of documents selected for judging together with all information required for evaluation: in our case, that means (1) the documents ids, (2) the relevance assessments, and (3) the inclusion probability for each document.

The inclusion probability π_k is simply the probability that the document k would be included in any sample of size m. In without-replacement sampling, $\pi_k = p_k$ when $m = 1$ and π_k approaches 1 as the sample size grows. For most common without-replacement sampling approaches, these inclusion probabilities are notoriously difficult to compute, especially for large sample sizes [8].

Additional judged documents, obtained deterministically, can be added to the existing sample with associated inclusion probability of 1. This is a useful feature as often in practice separate judgments are available; it matches perfectly the design of the Million Query Track pooling strategy, where for more than 800 topics a mixed pool of documents was created (half randomly sampled, half deterministically chosen).

Additionally, deterministic judgments may arise in at least two other natural ways: First when large-scale judging is done by assessors, it might be desirable to deterministically judge a given depth-pool (say the top 50 documents of every list to be evaluated) and then invoke the sampling strategy to judge additional

documents. (This strategy was employed in Terabyte 06 Track). Second, if it is determined that additional samples are required (say for a new run with many unjudged documents), one can judge either hand-picked documents and/or sampled documents and combine them with the original sample. Any collisions (where a document is sampled and separately deterministically judged) are handled by setting inclusion probability to 1.

4.3 Evaluation

Given a sample S of judged documents along with inclusion probabilities, we discuss here how to estimate quantities of interest (AP, R-precision, precision-at-cutoff).

For AP estimates, which we view as mean of a population of precision values, we adapt the generalized ratio estimator for unequal probability designs (very popular on polls, election strategies, market research etc.), as described in [13]:

$$\widehat{X} = \frac{\sum_{k \in S} v_k/\pi_k}{\sum_{k \in S} 1/\pi_k}$$

where v_k is the *value* associated with item k (e.g., the relevance of a document, a vote for a candidate, the size of a potential donation, etc.). For us, the "values" we wish to average are the precisions at relevant documents, and the ratio estimator for AP is thus

$$\widehat{AP} = \frac{\sum\limits_{rel(k)=1} \widehat{PC}(rank(k))/\pi_k}{\sum\limits_{rel(k)=1} 1/\pi_k} \tag{8}$$

where $\widehat{PC}(r) = \frac{1}{r} \sum\limits_{rank(k) \le r} \frac{rel(k)}{\pi_k}$ estimates precision at rank r and $k \in S$ iterates through *sampled documents only*).

Note that $\widehat{AP}$ mimics the well known formula $AP = \frac{sum\_of\_precisions\_at\_rel\_docs}{number\_of\_rel\_docs}$ because the numerator is an unbiased estimator for the sum of precision values at relevant ranks, while the denominator is an unbiased estimator of the number of relevant documents in the collection: $\widehat{R} = \sum_{rel(k)=1} \frac{1}{\pi_k}$. Combining the estimates for R and for precision at rank, $PC(r)$, we obtain also an estimate for R-precision:

$$\widehat{RP} = \widehat{PC}(\widehat{R}) = \frac{1}{\widehat{R}} \sum_{rank(k) \le \widehat{R}} \frac{rel(k)}{\pi_k}. \tag{9}$$

Finally, we note that the variance in our estimates can be estimated as well, and from this, one can determine confidence intervals in all estimates produced. Details may be found our companion paper [1].

4.4 Sampling strategy

There are many ways one can imagine sampling from a given distribution [8]. Essentially, sampling consists of a *sampling distribution* over documents (that should be dictated by the ranks of documents in the ranked lists and therefore naturally biased towards relevant documents) and a *sampling strategy* (sometimes called "selection") that produces inclusion probabilities roughly proportional to the sampling distribution. Following are our proposed choices for both the distribution and the selection algorithms; many others could work just as well. In the Million Query Track, due to unexpected server behavior, both the sampling distribution and the selection strategy were altered, yielding suboptimal chosen samples; nevertheless, we were able to compute the inclusion probability for each selected document and run the evaluation, though at a reduced accuracy and efficiency.

Sampling distribution (prior). It has been shown that average precision induces a good relevance prior over the ranked documents of a list. The AP-prior has been used with sampling techniques[5]; in metasearch (data fusion) [4]; in automatic assessment of query difficulty [2]; and in on-line application to pooling[3]. It has also been shown that this prior can be averaged over multiple lists to obtain a global prior over documents[5]. An accurate description together with motivation and intuition can be found in [5].

For a given ranked list of documents, let Z be the size of the list. Then the prior distribution weight associated with any rank r, $1 \le r \le Z$, is given by

$$W(r) = \frac{1}{2Z}\left(1 + \frac{1}{r} + \frac{1}{r+1} + \cdots + \frac{1}{Z}\right) \approx \frac{1}{2Z}\log\frac{Z}{r}. \tag{10}$$

We used for experimentation the above described prior, averaged per document over all run lists; Note that the our sampling strategy works with any prior over documents.

Stratified sampling strategy. The most important considerations are: handle *non-uniform* sampling distribution; *without replacement* so we can easily add other judged documents; *probabilities proportional with size (pps)* minimizes variance by obtaining inclusion probabilities π_k roughly proportional with precision values $PC_{rank(d)}$; and *computability of inclusion probabilities* for documents (π_k) and for pairs of documents (π_{kl}). We adopt a method developed by Stevens [8, 12], sometimes referred to as *stratified sampling*, that has all of the features enumerated above and it is very straight forward for our application. The details of our proposed sampling strategy can be found in [1]. Figure 3 provides an overall view of the statAP sampling and evaluation methodology.

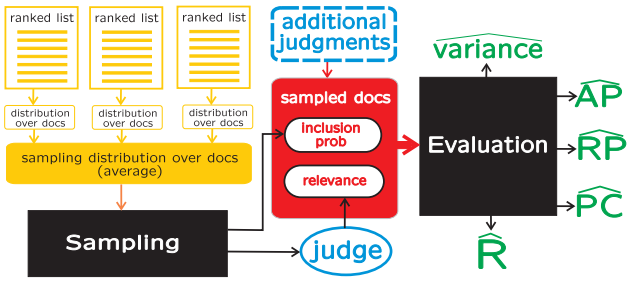

Figure 3: statAP: Sampling and evaluation design.

5 Alternation of methods

Half of the queries were served by alternating between the UMass algorithm and the NEU sample. The alternation was kept on separate "tracks", so that a judgment on a document served by NEU would not affect the document weights for UMass. If, say, NEU served a document that UMass had already served (and therefore that had already been judged), the judgment was recorded for NEU without showing the document to the user.

6 Statistics

The following statistics reflect the status of judgments as of October 16, 2007. Those are not the same judgments that were used by the track participants for their notebook papers, though the differences are small.

1,755 queries were judged
A total of 22 of those queries were judged by more than one person.
10 were judged by two people
5 were judged by 3 people
4 were judged by 4 people
3 were judged by 5 or more people

The actual assignment of topics to judging method was done in advance based on topic number. The following was the distribution of topics to methods:

443	of those used the UMass-only method
471	used the NEU-only method
432	alternated, starting with UMass
409	alternated, starting with NEU

Since assessors were shown a set of queries and could choose from them, we wondered whether there was an order effect. That is, did people tend to select the first query or not. Here is the number of times someone selected a query for judging based on where in the list of 10 it was.

	149 Terabyte		1MQ		
run name	unjudged	MAP	unjudged	$\mathcal{E}MAP$	NEU-MAP
UAmsT07MAnLM	65.23	0.0278	92.54	34.6649	0.0655
UAmsT07MTiLM	61.98	0.0392	91.03	47.4705	0.0956
exegyexact	76.81	0.0752	96.02	31.0603	0.0529
umelbexp	60.85	0.1251	91.28	95.8568	0.1474
ffind07c	28.98	0.1272	83.77	74.5032	0.1551
ffind07d	24.78	0.1360	83.26	77.5741	0.1633
sabmq07a1	21.18	0.1376	86.51	83.5744	0.1542
UAmsT07MSum6	32.21	0.1398	81.83	93.8634	0.1804
UAmsT07MSum8	23.79	0.1621	80.45	98.1381	0.2004
UAmsT07MTeVS	20.44	0.1654	81.95	85.1329	0.1822
hedge0	16.86	0.1708	80.85	109.4436	0.2217
umelbimp	14.65	0.2499	80.90	147.1392	0.2621
umelbstd	10.76	0.2532	82.62	148.3201	0.2663
umelbsim	10.38	0.2641	80.65	170.6247	0.2967
hitir2007mq	8.94	0.2873	80.31	150.2556	0.2844
rmitbase	7.52	0.2936	79.78	159.8542	0.3028
indriQLSC	6.31	0.2939	78.91	163.9249	0.3120
LucSynEx	12.30	0.2939	78.44	174.6228	0.3246
LucSpel0	12.36	0.2940	78.47	174.5095	0.3255
LucSyn0	12.36	0.2940	78.48	174.5020	0.3255
indriQL	6.56	0.2960	79.13	165.6138	0.3167
JuruSynE	8.04	0.3135	78.49	182.8071	0.3199
indriDMCSC	7.82	0.3197	79.61	162.8120	0.3049
indriDM	8.95	0.3238	80.28	165.9775	0.3132

Table 1: Performance on 149 Terabyte topics, 1692 partially-judged topics per UMass, and 1084 partially-judged queries per NEU, along with the number of unjudged documents in the top 100 for both sets.

Rank	1	2	3	4	5	6	7	8	9	10
Count	213	157	144	148	169	141	118	145	156	139
Percent	13.9%	10.3%	9.4%	9.7%	11.0%	9.2%	7.7%	9.5%	10.2%	9.1%

(The numbers add up to 1530 rather than 1755 because this logging was included partway through the judging process.)

Judgments came from the following sources:

1,478	NIST assessors
97	CIIR hired annotators
47	IR class project

The remaining judgments came from different sites, some (though not all) of which were participants. The number of judged queries ranked from 1 to 37 per site (other than those listed above).

7 Results

Table 1 shows evaluation results over the 149 Terabyte topics with the judgments from 2006 along with UMass and NEU results over the relevant judged queries from the 10,000. Also shown is the number of unjudged documents in the top 100 for each system.

Note that NEU could only evaluate over queries that were served fully or partially by the NEU sampling method; 429 UMass-only topics were skipped. Also, UMass did not exclude queries for which no relevant documents had been found; unjudged documents would be assigned probabilities of relevance that, while low, could allow recall to be estimated.

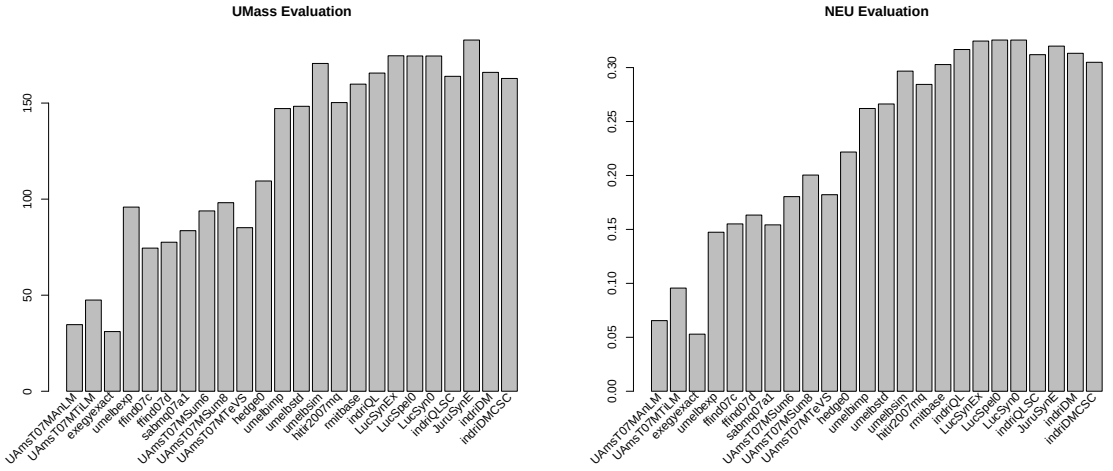

Figure 4: UMass and NEU evaluation results sorted by evaluation over 149 Terabyte topics.

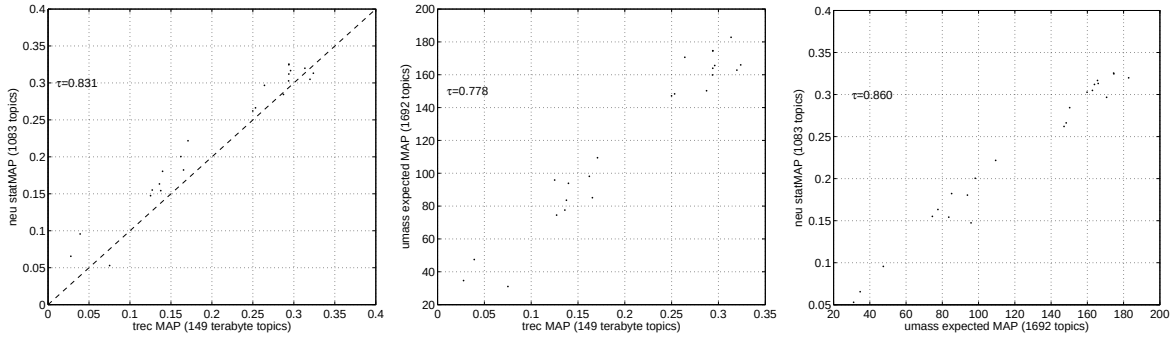

Figure 5: From left, evaluation over Terabyte queries versus NEU evaluation, evaluation over Terabyte queries versus UMass evaluation, and NEU evaluation versus UMass evaluation.

The UMass and NEU evaluations agreed with each other more than either one agreed with the evaluation over the 149 Terabyte topics. Kendall's tau rank correlation between UMass and NEU rankings was 0.860, while the correlation between the UMass ranking and the 149-queries ranking was 0.778 and the correlation between NEU and 149-queries was 0.831. None of the three agreed about the best system: indriDM was best over the 149 queries, but only 5th best by UMass and 6th best by NEU. JuruSynE was best according to UMass, but 3rd best by the Terabyte queries and 4th best by NEU. LucSyn0 was ranked 1st by NEU (by less than 0.00002 over LucSpel0), but 4th by UMass and 5th by the Terabyte queries.

As Figure 4 shows, the biggest disagreements between the two methods were about the ranks of umelbexp, umelbsim, and JuruSynE. umelbexp is not significantly different (at the 0.05 level) from ffind07c, ffind07d, sabmq07a1, or UAmsT07MSum6 by a two-sided paired t-test over the 149 Terabyte topics, suggesting that it is not incorrect for UMass to rank it above those 4 systems (though it is significantly different from UAmsT07MeTVS). The differences between JuruSynE, indriDMCSC, indriDM, indriQLSC, and indriQL are not significant, but the differences between JuruSynE and LucSpel0, LucSyn0, and LucSynEx are.

Overall, UMass agreed with 92.4% of the significant differences between systems as measured over the 149 Terabyte topics. NEU agreed with 93.7% of the siginificant differences. This difference is not significant by a one-sample proportion test ($p = 0.54$).

Figure 5 compares each of the three evaluations against each other. All three cases are highly correlated. The NEU method produces numbers that are close to TREC numbers over many more judgments, while the UMass method produces numbers on a different scale, but both are, for the most part, identifying the same relative differences between systems.

pair	confidence
exegyexact & UAmsT07MAnLM	0.9577
sabmq07a1 & UAmsT07MTeVS	0.7113
UAmsT07MSum6 & umelbexp	0.6639
umelbexp & UAmsT07MSum8	0.6920
umelbimp & umelbstd	0.6095
umelbimp & hitir2007mq	0.7810
umelbstd & hitir2007mq	0.6909
rmitbase & indriDMCSC	0.8412
indriDMCSC & indriQLSC	0.6552
indriDMCSC & indriQL	0.8480
indriQLSC & indriDM	0.7748
indriQL & indriDM	0.5551
LucSyn0 & LucSpel0	0.5842
LucSyn0 & LucSynEx	0.6951
LucSpel0 & LucSynEx	0.6809

Table 2: Probability that a difference in MAP is less than zero for selected pairs of systems.

As we described in Section 4, the UMass method is more interested in differences between pairs than in the value of $\mathcal{E}MAP$. For nearly all the pairs the confidence was 1, meaning that we predict that additional judgments will not change the relative ordering of pairs. Table 2 shows the confidence in the difference in $\mathcal{E}MAP$ for selected pairs that had less than 100% confidence. Note that many of the high-ranked systems (the indri set and the Luc set) were difficult to differentiate.

8 Analysis

umelbexp is an interesting case: for the 149 robust queries, on average only 39 of the documents it retrieved in the top 100 had previously been judged. Only three other systems retrieved fewer judged documents; 20 of the 24 systems retrieved 68 or more judged documents on average. But among those 39 retrieved, umelbexp had the highest judged-relevant-to-judged-retrieved ratio of any system, nearly 13% higher than the second highest. This suggests that the performance of umelbexp is being underestimated by the evaluation over 149 queries.

umelbexp suffered a similar fate in the judging for the new queries. Only 8 retrieved documents on average were judged, third lowest among all 24 systems, but it had the highest relevant-to-retrieved ratio of any system. It is not yet clear why so few documents were judged from umelbexp and the others.

Assessors made 33,077 judgments on the 801 alternating queries. Of these, 15,028 (45%) were chosen by both methods. 12,489 (38%) were chosen only by the UMass method, and 5,560 (17%) were chosen only by the NEU method.

Assessors re-judged 42 of the 149 Terabyte topics. Over the 2011 documents that were judged both times, agreement was 75%.

As extensively discussed in previous sections, 24 different runs were submitted to the Million Query track, where each run output a ranked list of documents for each one of $10,000$ queries. The ranked lists produced by all systems for a subset of $1,755$ queries were judged and their quality was assessed employing two different methodologies, UMASS and NEU. Each of the two methodologies evaluated the quality of the ranked lists for the $1,755$ queries by the means of some estimate of average precision (AP) and the overall quality of each system by some estimate of mean average precision (MAP), resulting into two test collections.

There are two questions that naturally arise: (1) How reliable are the given performance comparisons, and (2) how good are the test collections? We answer these two questions by employing Generalizability Theory (GT) [6, 7].

In particular, GT provides the appropriate tools to answers the question: To what extent does the variance of the observed average precision (AP) values reflect real performance difference between the systems as opposed to other sources of variance? During the first step of GT (the G-study), the AP value for a ranked

Effects	Variance	% of total variance
System Effect	0.0007	29.25%
Query Effect	0.0001	5.32%
S-Q Interaction Effect	0.0017	65.42%

Table 3: Variance components analysis based on 841 queries employing the UMASS methodology.

Effects	Variance	% of total variance
System Effect	0.0077	19.24%
Query Effect	0.0008	2.12%
S-Q Interaction Effect	0.0317	78.64%

Table 4: Variance components analysis based on 841 queries employing the NEU methodology.

list of documents produced by a single system ran over a single query can be decomposed into a number of uncorrelated effects (sources of variance),

$$AP_{aq} = \mu + v_a + v_q + v_{aq,err}$$

where μ is the grand mean over all AP values, v_a is the system effect, v_q is the query effect and $v_{aq,err}$ is the system-query interaction effect along with any other effect not being considered. Apart from the grand mean that is a constant, each of the other effects is modeled as a random variable and therefore it has mean and variance. In the same manner as the AP value decomposition, the variance of the observed AP value is decomposed into the corresponding variance components,

$$\sigma^2(AP_{aq}) = \sigma^2(a) + \sigma^2(q) + \sigma^2(aq, err)$$

Table 3 and Table 4 provide estimates of those variance components when the UMASS and the NEU methodology is employed, respectively. The figures in both tables are based on 841 queries (the common queries of UMASS and NEU test collections). Note that each variance component reported in the two tables along with the corresponding percentage is calculated on a per query basis. Therefore, 65.42% (or 78.64%) would be the percentage of the total variance that corresponds to the system-query interaction if a system runs on a single query when the UMASS (or NEU) methodology is employed.

While the G-study copes with the decomposition of variance of a single AP value into variance components due to a single system and a single query, the next step of GT (the D-study) considers the decomposition of the variance of the average of the AP values over all queries (MAP) into variance components due to a single system and a set of N_q queries. The variance components in the D-study can be easily computed by using the variance components computed during G-study as follows,

$$\sigma^2(Q) = \sigma^2(q)/N_q, \sigma^2(aQ) = \sigma^2(aq)/N_q$$

while the variance due to the system effect remains the same.

Table 5 and Table 6 provide the percentage of the variance of the MAP values that is due to the system effect, i.e. $\sigma^2(a)/(\sigma^2(a) + \sigma^2(q)/N_q + \sigma^2(aq)/N_q)$ for different number of queries (N_q) for the two methodologies. As can be observed, for both the UMASS and NEU methodologies, the variance in the MAP scores, in a test design of 841 queries (i.e. the design used in Million Query track) reflect the real performance difference between the systems, since the percentage of the total MAP variance that is due to the system effect is 99.71% and 99.50%, respectively. Therefore, any disagreement in the overall ranking of the systems by the two methodologies are due to the different estimators used by the two methodologies for computing AP and MAP values.

Number of Queries (N_q)	50	100	400	841
% of total variance due to system	95.38%	96.64%	99.40%	99.71%

Table 5: % of total variance due to system employing the UMASS methodology.

Number of Queries (N_q)	50	100	400	841
% of total variance due to system	92.25%	95.97%	98.96%	99.50%

Table 6: % of total variance due to system employing the NEU methodology.

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval, in part by the Defense Advanced Research Projects Agency (DARPA) under contract number HR0011-06-C-0023, in part by NSF grant IIS-0534482, and in part by Microsoft Live Labs. Any opinions, findings, and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsors.

References

[1] J. A. Aslam and V. Pavlu. A practical sampling strategy for efficient retrieval evaluation. Working draft available at the following URL: http://www.ccs.neu.edu/home/jaa/papers/drafts/statAP.html, May 2007.

[2] J. A. Aslam and V. Pavlu. Query hardness estimation using Jensen-Shannon divergence among multiple scoring functions. In G. Amati, C. Carpineto, and G. Romano, editors, *Advances in Information Retrieval: 28th European Conference on IR Research*, volume 4425 of *Lecture Notes in Computer Science*, pages 198–209. Springer-Verlag, 2007.

[3] J. A. Aslam, V. Pavlu, and R. Savell. A unified model for metasearch, pooling, and system evaluation. In O. Frieder, J. Hammer, S. Quershi, and L. Seligman, editors, *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, pages 484–491. ACM Press, November 2003.

[4] J. A. Aslam, V. Pavlu, and E. Yilmaz. Measure-based metasearch. In G. Marchionini, A. Moffat, J. Tait, R. Baeza-Yates, and N. Ziviani, editors, *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 571–572. ACM Press, August 2005.

[5] J. A. Aslam, V. Pavlu, and E. Yilmaz. A statistical method for system evaluation using incomplete judgments. In S. Dumais, E. N. Efthimiadis, D. Hawking, and K. Jarvelin, editors, *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 541–548. ACM Press, August 2006.

[6] D. Bodoff and P. Li. Test theory for assessing ir test collection. In W. Kraaij, A. P. de Vries, C. L. A. Clarke, N. Fuhr, and N. Kando, editors, *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 367–374. ACM Press, July 2007.

[7] R. L. Brennan. *Generalizability Theory*. Springer-Verlag, New York, 2001.

[8] K. R. W. Brewer and M. Hanif. *Sampling With Unequal Probabilities*. Springer, New York, 1983.

[9] B. Carterette. Robust test collections for retrieval evaluation. In *Proceedings of SIGIR*, pages 55–62, 2007.

[10] B. Carterette, J. Allan, and R. K. Sitaraman. Minimal test collections for retrieval evaluation. In *Proceedings of SIGIR*, pages 268–275, 2006.

[11] J. A. Rice. *Mathematical Statistics and Data Analysis*. Duxbury Press, second edition, 1995.

[12] W. L. Stevens. Sampling without replacement with probability proportional to size. *Journal of the Royal Statistical Society. Series B (Methodological), Vol. 20, No. 2. (1958), pp. 393-397.*

[13] S. K. Thompson. *Sampling*. Wiley-Interscience, second edition, 2002.