2005

Incremental Test Collections

Ben Carterette
University of Massachusetts - Amherst

James Allan
University of Massachusetts - Amherst

Incremental Test Collections

Ben Carterette and James Allan
{carteret,allan}@cs.umass.edu
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003

ABSTRACT

Corpora and topics are readily available for information retrieval research. Relevance judgments, which are necessary for system evaluation, are expensive; the cost of obtaining them prohibits in-house evaluation of retrieval systems on new corpora or new topics. We present an algorithm for cheaply constructing sets of relevance judgments. Our method intelligently selects documents to be judged and decides when to stop in such a way that with very little work there can be a high degree of confidence in the result of the evaluation. We demonstrate the algorithm's effectiveness by showing that it produces small sets of relevance judgments that reliably discriminate between two systems. The algorithm can be used to incrementally design retrieval systems by simultaneously comparing sets of systems. The number of additional judgments needed after each incremental design change decreases at a rate reciprocal to the number of systems being compared. To demonstrate the effectiveness of our method, we evaluate TREC ad hoc submissions, showing that with 95% fewer relevance judgments we can reach a Kendall's tau rank correlation of at least 0.9.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]; H.3.4 [**Systems and Software**]: Performance Evaluation

General Terms

Algorithms, Experimentation, Measurement

Keywords

information retrieval, evaluation, test collections, algorithms

CIKM'05, October 31–November 5, 2005, Bremen, Germany.

1. INTRODUCTION

System-based information retrieval evaluation requires a *test collection*: a set of items to search (a *corpus*), a set of *topics* from which we can extract *queries*, and a set of *relevance judgments* that tell us which documents are relevant to which topics [12]. The test collections put together by NIST for the Text Retrieval Conferences (TREC) are well-known and widely used in the IR community [13]. Each TREC collection consists of a corpus of mostly news articles, 50 subject-based topics, and tens of thousands of relevance judgments made by NIST.

Researchers may want to put together their own test collections. They may have acquired a new corpus or created a new set of topics. Search engine designers in a practical setting have a set of documents that will be searched and can solicit topics from the people that will be using the system. Corpora are readily available; the problem is that relevance judgments are very expensive. Researchers must hire annotators to read documents and decide whether they are relevant to each topic. Judging every document in a large corpus is impossible. Judging only a subset raises questions about how well the subset represents the document space and how well it will generalize to new systems [5, 14]. NIST has addressed this by judging a large subset of documents retrieved by actual retrieval systems, but most developers and researchers do not have the resources to judge the tens of thousands of documents that NIST judges. We need a way to minimize the cost of relevance judgments but still have confidence in the result of an evaluation.

Retrieval system design is typically an iterative process: we make design decisions based on the results of evaluations of previous systems. We can exploit the iterative nature of the design process to construct sets of judgments incrementally. At each step we judge only the documents that are most important to the evaluation at that step. Of course there is a tradeoff in that we will have less confidence in the result of a comparison because the judgments are not complete, but after each iteration we add more documents to the collection and thus gain confidence in the result.

In section 2 we describe an algorithm that produces a set of relevance judgments while comparing two systems. In section 3 we present results of comparisons of pairs of systems submitted to the TREC ad hoc tracks, showing that with as little as two judgments per topic we can correctly identify significant differences in over 93% of cases. In section 4 we describe how our algorithm generalizes to an evaluation of multiple systems. In section 5, the main experimental re-

sults are presented. We simulate the iterative design process to show how the number of necessary judgments increases as new systems are developed. We also use our algorithm to rank TREC systems, achieving a Kendall's tau correlation greater than 0.9 with 95% fewer judgments than done by NIST. In sections 6 and 7 we analyze the algorithm and compare it to previous work. We conclude in section 8.

2. ALGORITHM DESCRIPTION

Given two systems, the algorithm is to identify documents that inform us most about the difference between them, and to decide when we have enough information to stop judging. We use heuristics that have intuitive appeal; we have not yet formally proved anything about this algorithm.

2.1 Identifying Important Documents

Intuitively, a document that appears at the same rank in two lists provides no information about the difference between the systems, while a top-ranked document in one system that is not ranked at all by the other may provide a lot of information.

Average precision (AP) of a ranked list is the sum of the precisions at each rank at which a relevant document was retrieved, divided by the total number of relevant documents known for the topic.

$$AP = \frac{1}{|R|} \sum_{d \in R} prec@rank(d)$$

AP is computed for a ranked list of a single topic. Mean average precision (MAP) is the average of the APs for each topic in the test collection. MAP is a standard evaluation measure, and is known to be stable [2].

Unjudged documents are usually assumed to be nonrelevant for the purpose of computing AP. We could calculate AP using some incomplete set of judgments R, judge an unjudged document, and recalcuate AP with the new set of judgments R'. The difference in the two calculations is the *effect* on AP of judging that document. Applying our intuition, we want to find the document that maximizes the difference in two MAP scores. That document is the one that, if relevant, would have the greatest difference in its effects on the APs of the topics for which it was retrieved by both systems.

When we begin, there are no relevance judgments and average precision for each topic is 0. The first relevant document we find has an effect of $1/r_1(d)$ on AP_1 and $1/r_2(d)$ on AP_2, where $r_i(d)$ is the rank of document d in ranked list i. Since we want to find the greatest *difference* in effects, we initially assign each document a weight $w_d = |\frac{1}{r_1(d)} - \frac{1}{r_2(d)}|$. Documents are presented to an assessor for judging in nonincreasing order by weight, ignoring topic.

When there are some relevance judgments, the effect a document might have on AP is dependent on other relevant documents found for that topic. For example, suppose documents at ranks 1, 2, 4, and 5 in ranked list 1 are relevant, nonrelevant, nonrelevant, and relevant, respectively, and the document at rank 3 is unjudged. Then $AP_1 = \frac{(1/1+2/5)}{2}$. If we subsequently judged the document at rank 3 to be relevant, the new AP would be $AP_1' = \frac{(1/1+2/3+3/5)}{3}$. The effect of the judgment on the numerator of AP_1 is $3AP_1' - 2AP_1|R| = (1/1+2/3+3/5) - (1/1+2/5) = 2/3+1/5$. After a similar calculation for ranked list 2, the difference in effects is |(effect on AP_1) - (effect on AP_2)|.

In general,

$$w_d = \left| \left(prec@r_1(d) + \frac{1}{r_1(d)} + \sum_{\substack{d' \in R_j \\ r_1(d') > r_1(d)}} \frac{1}{r_1(d')} \right) - \left(prec@r_2(d) + \frac{1}{r_2(d)} + \sum_{\substack{d' \in R_j \\ r_2(d') > r_2(d)}} \frac{1}{r_2(d')} \right) \right|$$

where R_j is the set of relevant documents after j documents have been judged. The effect on AP_i of finding d relevant is the precision at the rank of d plus the reciprocal rank of d (to account for d being assumed relevant) plus the reciprocal rank of every relevant document that follows d in the ranking; the weight of document d is the difference in effects.

2.2 Stopping Condition for Topics

After finding a relevant document for topic t, we may, if it seems clear that the systems are quite different, want to stop judging documents from that topic. We calculate the average precision of both systems' ranked lists for topic t. If one is greater than the other, we want to guess how likely it is that the other can catch up given more judgments. If it is unlikely, we can stop.

We calculate a lower bound ℓ_t on the number of judgments it would take to catch up. Assuming a best-case scenario in which every unjudged document retrieved by the worse system will be judged relevant, we count from the top-ranked unjudged document down until the hypothetical average precision of the worse system is equal to or greater than the average precision of the better system. We know it will take at least that many documents to catch up. If that number is high, it is unlikely that the lists are equal and we can stop judging documents for the topic; if it is low, we must continue. When we stop judging for a topic, we flag that topic "done".

The algorithm takes as a parameter a cutoff c. If the lower bound $\ell_t > c$, we stop judging documents from that topic. The choice of cutoff reflects a tradeoff between costs of judgments and costs of accuracy errors. As we shall see, a higher cutoff results in greater accuracy but more judgments. The cutoff can also be thought of as the point at which you believe the probability of judging c consecutive documents relevant is nil.

2.3 Stopping Condition for Systems

If it becomes obvious in the course of judging documents that one system is far better than the other, we may want to stop judging early. We perform a one-tailed sign test using the "done" topics; if the difference is significant, we stop judging documents altogether.

Our use of the sign test is a heuristic, and the result of the test should not be taken to mean the difference actually is significant, except to the degree that we correctly predict significance. Note that our use of the sign test makes the assumption that topics that are tied or not yet "done" provide no information. This is a strong assumption. We might make a weaker assumption, e.g. that any topic that is not "done" is equally likely to go to either system.

TREC	topic numbers	no. runs	judg-ments	no. rels	% signif
TREC-3	151-200	40	97,319	9,805	79.2
TREC-4	202-250	33	87,069	6,503	75.0
TREC-5	251-300	61	133,681	5,524	71.5
TREC-6	301-350	74	72,270	4,611	75.1
TREC-7	351-400	103	80,345	4,674	77.8
TREC-8	401-450	129	86,830	4,728	77.1

Table 1: TREC corpora. "% signif" is the percent of pairs with a significant difference by the sign test.

3. EXPERIMENTS

We downloaded submissions to the ad hoc tracks of TRECs 3 through 8. Table 1 shows the number of runs, topic numbers, number of relevance judgments and relevant documents, and percent of pairs significantly different by the sign test for each TREC.

3.1 Implementation

Because there are many pairs of systems (e.g. 40 systems submitted to TREC-3 gives 780 possible pairs), we used some approximation in the implementation to save computation time. We only update the weights after every 50 relevant documents. We only judge documents that are ranked between 1 and 100 by either system. We only calculate the lower bound for a topic when we find a relevant document in that topic, but because a nonrelevant judgment may increase the lower bound, we also calculate the lower bound for every topic after every 25 judgments. For TRECs 3 and 4 we compared every pair of systems using our algorithm. For TRECs 5 through 8 we picked 25% of the pairs (or 1000, whichever is greater) at random.

3.2 Evaluation

The algorithm produces a set of relevance judgments. To evaluate the set, we want to know:

1. How often does it indicate a significant difference when a significant difference exists (by the NIST judgments)? We think of this as "recall of significant pairs".
2. How often does it correctly identify the relative difference when it indicates a significant difference? This is "precision of significant pairs."
3. How often does it correctly identify the relative difference regardless of whether it indicates a significant difference? This is "overall accuracy."
4. How often does it identify a difference as significant when it is not truly significant? These are "false alarms."

3.3 Results

The results are shown in Figure 1. Each point in the figure is the average over more than 6500 pairs of systems from all TRECs at a given cutoff. We see that as the number of judgments increases (by increasing the cutoff), precision, recall, and accuracy increase, while false alarms decrease (1-FA increases). With only 59 judgments, an average of just over one per topic, we can correctly identify a significant difference when one exists 82% of the time, and we correctly identify the sign of the difference 93% of the time. Table 2 shows the numbers. "Acc" is the overall accuracy; remember that it includes pairs with a nonsignificant difference.

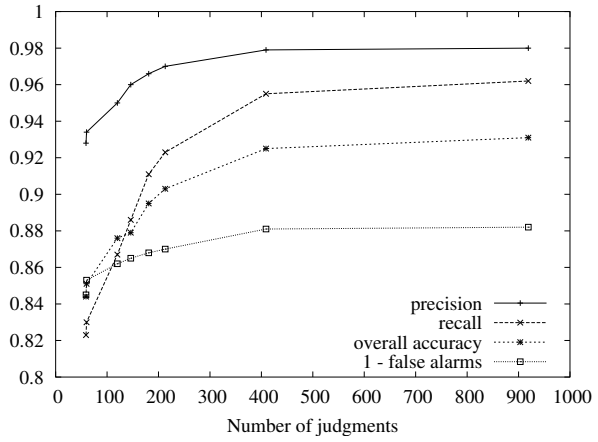

Figure 1: Number of judgments vs. precision and recall as cutoff increases. Points are, from left to right, cutoffs $c = 0, 1, 2, 3, 4, 5, 10, 20$.

c	prec	rec	acc	false	judgments	rels
0	92.8	82.3	84.4	15.5	59	20
1	93.4	83.0	85.1	14.7	60	20
2	95.0	86.7	87.6	13.8	120	52
3	96.0	88.6	87.9	13.5	146	68
4	96.6	88.2	89.5	13.2	181	88
5	97.0	92.3	90.3	13.0	213	106
10	97.9	95.5	92.5	11.9	409	199
20	98.0	96.2	93.1	11.8	919	405

Table 2: Results by cutoff averaged over pairs of systems from all TRECs.

The false alarm rate is relatively high; this is because of the strong assumption we make about topics that are not "done".

For comparison, Table 3 shows results using mean average precision calculated using a pool of depth k, which is the NIST approach. Note that it requires many more relevance judgments to achieve the same rate of differentiation, though the traditional measures are more accurate. Compare $k = 5$ to our method with cutoff 10: with approximately the same number of judgments, the accuracy on significant pairs is the same, but our method finds many more significant pairs. (Recall and accuracy on a pool of depth 100 fall short of 1.0, which may be surprising considering that NIST uses pools of depth 100. In this case, the pool is only formed from the two systems being compared, not the full set of systems, so some relevant documents are not found.)

Table 4 shows results at cutoff $c = 4$ broken out by TREC. Note that more documents are judged in earlier TRECs than in later. We believe that this is because systems from later TRECs overlap more in the documents they retrieve. As systems get more similar, the weights of documents decrease, and it becomes more clear exactly which documents need to be judged to differentiate between two systems.

The number of judgments produced is influenced by a number of factors, including the number of relevant documents in the topic, the number of relevant documents re-

k	prec	rec	acc	false	judgments	rels
1	97.0	62.8	83.5	7.9	93	45
5	97.5	77.2	87.9	8.7	444	190
10	98.0	81.4	89.7	8.2	867	336
20	98.7	85.3	91.6	7.5	1700	570
50	99.5	89.1	93.0	5.9	4127	1077
100	99.8	91.5	94.9	5.1	7993	1653

Table 3: Evaluation by AP calculated using a pool of depth k averaged over pairs of systems from all TRECs.

TREC	prec	rec	acc	false	judgments	rel
TREC-3	97.3	90.1	90.5	11.2	170	99
TREC-4	94.9	87.8	85.2	13.3	222	101
TREC-5	97.0	92.0	91.4	14.3	192	85
TREC-6	97.8	91.4	89.7	12.8	179	80
TREC-7	96.7	90.7	90.0	13.0	166	83
TREC-8	96.1	91.7	89.0	13.7	158	81

Table 4: Results by TREC for cutoff $c = 4$.

trieved by both systems, the percent difference in average precision in the two systems, and the similarity (in terms of documents retrieved) between the two systems. The algorithm judges documents from every topic, on average. The Pearson correlation between predicted MAP and the "true" MAP calculated using the NIST judgments ranges from 0.44 to 0.87 (when $c = 0$ and $c = 20$ respectively). The correlation between the difference in predicted MAP of both systems and the difference in NIST MAP ranges from 0.75 to 0.96 ($c = 0$ and 20 respectively), indicating that this algorithm is better at predicting the difference between two MAPs then actually predicting MAP.

These results show that the algorithm can successfully find a difference between two systems when one exists with a small number of judgments. Next we want to know how many additional judgments it would take as more systems are developed.

4. COMPARING MULTIPLE SYSTEMS

Information retrieval system design is typically iterative, with design decisions at step n influenced by evaluations of the previous $n-1$ systems. Our algorithm leaves uncertainty in the results of previous evaluations, so we do not want to assume that an evaluation was correct and never look at it again. We want to reevaluate systems as we accumulate more data. As we repeatedly compare any two systems, if the result is the same each time, it becomes less likely that there is an error in the evaluation. So each time we develop a new system, we want to produce a set of relevance judgments that we can use to compare all S existing systems simultaneously.

We view the comparison as $O(S^2)$ simultaneous pairwise comparisons, and generalize as follows: the document weight is the maximum of its weight in each pair. As before, we sort documents by weight. We stop judging a topic when the ranking of lists by average precision is unlikely to change, and stop judging altogether when the ranking of lists by mean average precision is unlikely to change. This is described in more detail below.

4.1 Identifying Important Documents

The document most likely to tell us something about the ranking of systems is the one that provides the most information about the differences between each pair of systems. That suggests using the average weight of a document in all pairs. The problem with using the average is that a document that has low weight in a few pairs may have a relatively high average weight compared to one that has high weight in a few pairs but low weight in most pairs. These documents are poor discriminators, but will be judged first.

Instead of average weight, we use the maximum weight of a document in any pair of systems it appears in: $w_d = \max_{i<j} |\frac{1}{r_i(d)} - \frac{1}{r_j(d)}|$. This way we are guaranteed to judge documents that are ranked at the top of a list and that will also discriminate at least one pair of systems. Since many documents will have the same maximum weight, we use the average weight as a tiebreaker. We update weights as in section 2.1.

4.2 Stopping Condition for Topics

We want to stop judging a topic when the ordering of ranked lists by average precision is unlikely to change. We assign to each pair of ranked lists (t_i, t_j) a probability of swapping: $P_{swap}(t_i, t_j) = 0$ if $\ell_t > c$, or $P_{swap}(t_i, t_j) \propto \exp(-\ell_t^2)$ if $\ell_t \leq c$, where ℓ_t is the lower bound on judgments to catch up as described in section 2.2. If additional judgments would not change the ranking significantly, we can accept some uncertainty because it will get averaged out in the mean average precision calculation. We calculate the expected rank correlation between our current ranking and a future ranking given the probabilities of each pair swapping. If the expected rank correlation is high, we do not expect the ranking to change much, so we stop judging the topic. We set the correlation cutoff at 0.9 to reflect Voorhees' conjecture that that is the highest correlation that we should expect to achieve simply because of inter-annotator disagreement [11].

4.3 Stopping Condition for Systems

We likewise stop judging documents when the system ranking is unlikely to change. Each pair of systems (s_i, s_j) is assigned a probability of swapping: $P_{swap}(s_i, s_j) = 0$ if the systems have been found to be significantly different (using the sign test as in section 2.3); if not, we calculate the probability that one system can catch up to the other, with $P_{swap}(s_i, s_j) = 0.5$ if the number of topics favoring s_i equals the number favoring s_j, and $P_{swap}(s_i, s_j) = P(T \geq 25 - T_j)$ where $T \sim \text{Binom}(50 - (T_i - T_j), 1/2)$, i.e. the probability that s_j will accumulate enough topics in its favor (among the ones not yet decided) to have a majority. With those probabilities we calculate $E[\tau]$ and stop if it is greater than 0.9.

Voorhees previously used a definition of P_{swap} that was calculated using mean average precision differences on multiple sets of relevance judgments [10]. Since our judgments are incomplete, and we only have one set, we cannot use the same definition.

5. EXPERIMENTS

The first experiment is to verify that the algorithm can produce a set of judgments that correctly rank TREC sys-

tems. We take the full sets of systems submitted to TRECs 3, 4, 5, and 6 and run the algorithm with cutoffs ranging from 0 to 20.

For the second experiment, we wish to simulate the iterative design process. We do not know what a typical design process looks like, and it is not obvious how to simulate the process. Options we considered are:

1. Simulate different systems by running our in-house retrieval engine with different parameter settings. One problem with this is that it does not provide much variance in systems. Another is that it would be hard for other researchers to duplicate our experiments.
2. Use TREC systems from a single site. There are only a few systems from each site, and they frequently differ only in whether they use title, description, or narrative queries. Again, there is not a lot of variance.
3. Use all TREC systems, starting with two chosen randomly and adding an additional randomly selected system at each iteration.

The experiment we decided on is selecting S systems at random, starting at $S = 2$ and working up. Each evaluation of S systems is independent of any other evaluation. We expect that on average this simulates the iterative design process.

An additional advantage of selecting random subsets of systems is that we do not make the mistake of designing our algorithm so that it performs well only on full sets of TREC systems, which are comparatively easy to rank correctly.

For each TREC we ran the algorithm at least 100 times with S randomly selected systems, for more than 600 total algorithm runs for each S up to 60. We used cutoff $c = 4$ to keep the computation time low.

5.1 Implementation

Finding the maximum weight document is computationally expensive: $O(S^2D^2 \ln D)$, where D is the number of unique documents ranked by all S systems and assuming sorting documents by weight is in $O(D \ln D)$. To reduce computation time, we recalculate the weights after every 25 relevant documents are found. We only judge documents that appear above rank 50 in at least one system (meaning that we can never do better than a set of judgments formed from a pool of depth 50). If there are more than 40 systems, we find the maximum weight in a random sample of pairs.

5.2 Evaluation

A comparison among a set of systems produces a set of relevance judgments. We rank systems by mean average precision calculated using that set. To estimate the quality of the ranking, we calculate its correlation to the ranking of systems by MAP calculated using the full set of judgments. We use the rank correlation measure *Kendall's tau* [6], which has become a *de facto* standard in this type of study.

Kendall's tau is a function of the number of concordant and discordant pairs between rankings: $\tau = \frac{C-D}{\binom{n}{2}}$. It ranges from -1 if the two lists are inverted to 1 if they are identical. It is 0 if there is no correlation. If we know the τ correlation, we also know that the percentage of pairs that were correctly differentiated is $.5(1 + \tau)$.

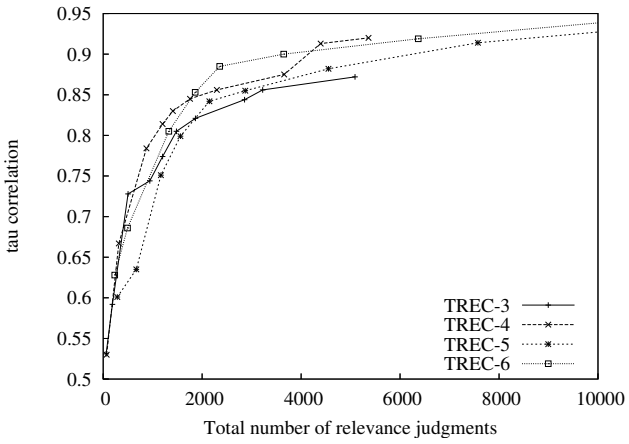

Figure 2: Number of judgments vs. tau correlation for complete sets of TREC systems. Points from left to right are cutoffs $c = 0, 1, 2, 3, 4, 5, 7, 10, 15, 20$**.**

5.3 Results

5.3.1 Full sets of TREC systems

Figure 2 shows the result of using our algorithm to produce judgments for evaluation of TREC systems. High correlations are achieved with very small sets of judgments. A set of 2000 judgments is less than one judgment per topic per system, but that is enough to achieve $\tau \approx .85$ on the TREC-6 collection.

TREC-3 did not quite achieve $\tau \geq 0.9$ in any of our experiments, but with 95% fewer judgments than done by NIST, we have $\tau = 0.87$. For TREC-4, we achieve $\tau > 0.9$ with 95% fewer judgments than done by NIST. TREC-5 is a bit harder, but we achieve $\tau > 0.9$ with 94% fewer judgments. For TREC-6, with 95% fewer judgments we achieve $\tau = 0.9$. In general, tau correlation and judgments increase as cutoff increases.

Table 5 shows the numbers for each set of TREC systems. Compare the TREC-6 numbers to Table 6, which shows numbers for different pool depths. Pooling is surprisingly effective, with a correlation of .82 with a depth of only 1! Although shallow pools work well with the TREC-6 corpora, it does not necessarily follow that a shallow pool would be sufficient for any corpora. For the TREC-3 systems, for instance, a pool of depth 1 results in a correlation of only .71 with 973 judgments.

5.3.2 Subsets of TREC systems

Figure 3 shows the size of the test collection produced by the algorithm increasing as systems are incrementally designed and added to the set. Accuracy was about $\tau = 0.77$, which implies that 88.4% of the pair differences were correctly identified—consistent with the 89.5% shown in Table 2. With two systems, we make about 200 judgments, also consistent with Table 2. When we add a third system, we nearly triple the number of judgments—with more than two systems, it is less likely that a single document will be retrieved at the same rank by all systems. Subsequent systems add fewer and fewer judgments to the set. Figure 2, in which number of judgments increases with cutoff, suggests

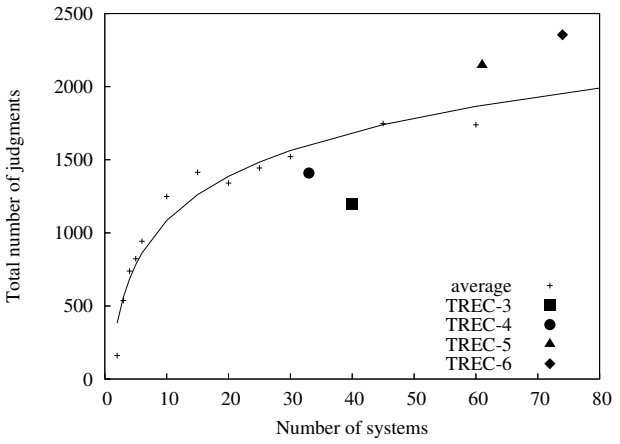

Figure 3: Number of systems simultaneously compared vs. number of judgments made when $c = 4$. Rank correlation remains approximately constant at $\tau = 0.77$.

that higher values of c would result in approximately parallel curves higher up. The fit curve is $y = a \log S$, which implies that the rate of increase in the number of judgments is inversely proportional to the number of systems being compared.

Figure 3 also shows the size of the test collections produced by simultaneous comparisons of the 40, 33, 61, and 74 systems from TRECs 3, 4, 5, and 6 respectively at $c = 4$. They are close to the fit log function. They have higher correlations than the random samples, probably because there is more overlap between systems.

The correlation between number of judgments made for each topic and number of true relevant documents in each topic is 0.47, indicating that more documents are judged from topics that have more relevant documents. A correlation of 0.76 between NIST MAP and number of documents judged from each system indicates that more documents are judged from systems that are better. The correlation between predicted MAP and NIST MAP is 0.81. We expect these correlations to increase at higher cutoffs.

6. ANALYSIS

The results are somewhat unbelievable. With an average of just over one judgment per topic (at cutoff 0) we can distinguish between 82% of the systems that have a significant difference, and we get 93% of the differences we find correct. Increasing to only eight judgments per topic (at cutoff 10) we distinguish between 93% of the systems with a significant difference, and we get 98% of comparisons correct. How is that possible?

In general, one system will be better than another if it tends to retrieve more relevant documents at higher ranks. Mean average precision is higher in that case, as are most evaluation measures. Consider cutoff $c = 0$ and a simplified case in which the two systems do not overlap at all in retrieved documents. When we sort documents by weight, the top-ranked documents by both systems in all topics will be tied with weight $w_d = 1$. When we begin to judge, if the better system ranked more relevant documents first, there is a higher probability that when we find a relevant document it is from the better system. If one system is better on a majority of topics, there is a high probability that we will find more relevant documents from that system, and if the majority is enough to be significant, it is likely that we will discover that quickly.

Of course we can imagine circumstances under which the algorithm would fail, but if it is correct on x out of T topics on average and on the other $T - x$ it is essentially random, the expected number of topics correctly sorted is $\frac{x+T}{2}$. If $x = 20$ and $T = 50$, the algorithm will be right on a majority of topics.

In light of the results for single pairs, the results for multiple comparisons make sense. If the accuracy at $c = 0$ is 84%, the expected τ is .688, which is a bit higher than we observe in TREC collections due to overlap in retrieved sets. Accuracy at $c = 10$ is 92.5%, implying a τ of .85, a bit lower than we observe, again due to overlap. Overlap affects pairwise comparisons as follows: at low cutoff, more overlap means more error, because judging a single document from one pair will "decide" any pair that document occurs in. At high cutoff, more overlap means less error, because more documents are judged from each pair before the pair is completed.

Iteratively comparing the systems gives more confidence in the results of each individual pair. It results in the same pairs being compared over and over again, so if the result of the comparison is the same each time, it becomes less and less likely that it is a mistake. Additionally, the size of the test collection keeps increasing, and more judgments always means more confidence.

7. PREVIOUS WORK

The traditional means of obtaining a set of relevance judgments is by system pooling, as we mentioned in section 1. At TREC, NIST pools the top N documents retrieved for each topic by each system and judges the entire pool. Usually $N = 100$; sometimes $N = 50$ or 200. Table 1 shows the number of relevance judgments this entails. If an assessor can make one judgment every 30 seconds, the 72,270 judgments collected for TREC-6 would take 25 days of around-the-clock work to produce—and that is for only 0.26 percent of the 27.8 million judgments that could be obtained for 50 topics on the entire 556,000-document collection. Even this small set is infeasible for most researchers.

Although the pooling method results in a small subset of the total number of possible judgments, it is sufficient for research purposes [14, 10]. This suggests that it might not be necessary to pool 100 documents. Pools of depth 20, 10, or even 5 result in good correlation to the NIST ranking of TREC systems by MAP with a pool of depth 100 (Table 6).

Another option is to construct topics such that only a subset of the collection could be relevant. An example is restricting topics to events that happened on a certain date. Another example is known item retrieval, in which topics are defined to have only one relevant document [1]. These sorts of topics do not provide enough variance to allow us to make general statements about differences between systems [9].

Soboroff et al. investigated a way to construct test collections with no judgments at all [8]. Using a model of how relevant documents occur in pools, they randomly selected documents from a pool and assigned relevance to all the chosen documents. Ranking systems by MAP calculated using

	TREC-3			TREC-4			TREC-5			TREC-6		
c	tau	judgments	rels	tau	judgments	rels	tau	judgments	rels	tau	judgments	rels
0	.531	60 (99.9%)	50	.530	70 (99.9%)	50	.601	285 (99.8%)	118	.628	234 (99.7%)	93
1	.592	186 (99.8%)	125	.667	316 (99.6%)	175	.635	670 (99.5%)	250	.686	491 (99.3%)	200
2	.728	502 (99.5%)	325	.784	874 (99.0%)	450	.751	1166 (99.1%)	450	.805	1327 (98.2%)	475
3	.744	942 (99.0%)	575	.814	1201 (98.6%)	600	.799	1562 (98.8%)	600	.853	1860 (97.4%)	675
4	.774	1201 (98.8%)	675	.830	1409 (98.4%)	625	.842	2147 (98.4%)	850	.885	2355 (96.7%)	825
5	.762	1545 (98.4%)	900	.845	1755 (98.0%)	850	.855	2872 (97.9%)	1075	.900	3649 (95.0%)	1300
7	.821	1864 (98.1%)	1025	.856	2298 (97.4%)	1100	.882	4554 (96.6%)	1615	.919	6367 (91.2%)	1800
10	.844	2858 (97.1%)	1500	.875	3657 (95.8%)	1575	.914	7573 (94.3%)	2213	.955	13060 (81.9%)	2391
15	.856	3224 (96.7%)	1625	.913	4395 (95.0%)	1900	.944	13067 (90.2%)	2933	.967	15426 (78.7%)	2928
20	.872	5090 (94.8%)	2455	.920	5362 (93.8%)	2150	.953	16766 (87.5%)	3237	.976	24187 (66.5%)	3706

Table 5: Results for evaluations of TRECs 3–6 at various cutoffs. The number in parentheses is percent decline from full NIST judgment count.

depth	tau	judgments	rels
1	.820	1747	460
5	.899	6652	1216
10	.930	12209	1747
20	.964	22937	2477
50	.981	52874	3575

Table 6: Total number of judgments in a pool of depth k and tau correlation to the NIST ranking of TREC-6 systems.

these so-called "pseudo-rels", they achieved a rank correlation of between .4 and .6 with the NIST ranking.

Sanderson and Joho judged documents from a single system (or a subset of systems) and, when the system was a good one, achieved high rank correlations [7]. They also used successive relevance feedback runs to incrementally add documents to the pool, and achieved a high rank correlation with that method as well. These methods emulate "Interactive Searching and Judging", described in more detail below. They require many more judgments than our algorithm.

Results of experiments with shallow pools, no pools, or single-system pools suggest to us that TREC systems are getting more similar in terms of the sets of documents they retrieve. Further evidence is that the average number of documents a system contributes to the pool has decreased. We could imagine that retrieval systems that use very different algorithms to retrieve from a very large corpus might overlap very little in the documents they retrieve, and in that case we would not expect any of these methods (excluding a pool of reasonable depth) to provide a stable evaluation.

A different way to construct a test collection called "Interactive Searching and Judging" (ISJ) was presented by Cormack et al. [4]. In ISJ, users submit queries to a retrieval system and judge retrieved documents, formulating and submitting new queries as they learn about the topic and corpus. They found that with a few hours of work, annotators could produce as many relevant documents as exist in the official set of judgments. Interestingly, there was only 33% overlap between the ISJ set and the official set, but the evaluation results were nearly the same [10]

Although Cormack et al. convincingly show that ISJ works with the TREC-6 corpus, whether it would work with a larger corpus is an open question. ISJ does not necessarily judge documents from any submitted run, which means it is conceivable, though perhaps not likely, that the set of judgments it produces does not intersect with any system.

Sanderson and Joho conclude that their simulation of ISJ confirms that it works, but our belief is that their results are due to TREC systems getting more similar. In TREC-7 and TREC-8, more than half the systems retrieved 60% or more of the relevant documents. This suggests not that systems are getting better at finding relevant documents, but that they are finding the same relevant documents, and as a result, the judged pool is a smaller sample of the full corpus. Judging documents from one or several decent systems will, therefore, result in an evaluation with high rank correlation simply because it is very likely that the other systems retrieved the same documents, and if they did not, they probably did not retrieve very many relevant documents.

"Move-to-Front pooling" (MTF) was proposed by Cormack et al. [4]. It imposes a priority ordering on the pool based on the assumptions that higher-ranked documents are more likely to be relevant and documents from systems that have recently discovered a relevant document are more likely to be relevant. MTF discovers relevant documents faster than traditional pooling, and achieves high positive rank correlations with much less work. Zobel proposed a similar method [14] that first judges a shallow pool, then, based on the results of the evaluation, extrapolates which systems and topics are likely to provide more relevant documents, and extends the pool using more documents from those. Zobel also discovers relevant documents faster than traditional pooling.

It is worth considering the performance of MTF and our algorithm on two extreme cases.

1. Consider two systems that retrieve two mutually exclusive sets of documents. In this case, our algorithm's weights are equal to the reciprocal ranks of each document, and therefore the ordering by weight is identical. MTF is likely to be slightly superior in this case, because it makes an additional assumption that if a system is good on one topic, it is likely to be good on other topics.
2. Consider two systems that are identical. In our algorithm, every document has weight 0. The algorithm does not have to judge any documents to know the systems are equivalent. MTF has no way of knowing that.

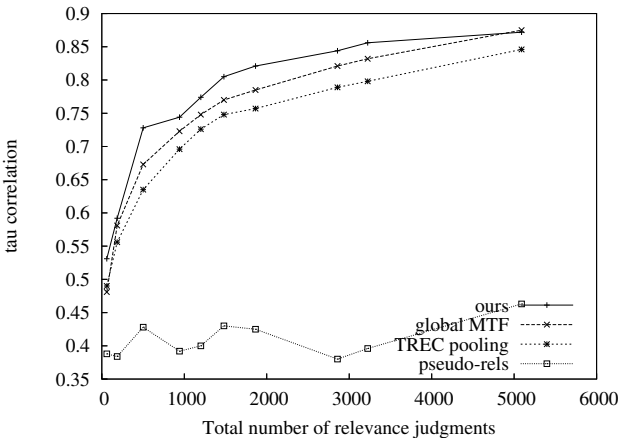

Figure 4: A comparison of four algorithms for producing test collections from TREC-3 systems.

Figure 4 shows a comparison of our algorithm, MTF pooling, TREC pooling, and Soboroff et al.'s pseudo-rels. The curve representing our algorithm is the same as the one for TREC-3 in Figure 2. For the other methods, we created a pool of documents of the same size as the one our algorithm produced, so that we could directly compare the performances on test collections of the same size. We averaged over trials of each method with random topic orderings. Our method outperforms all the others, though MTF catches up at the final point in the curve. Our algorithm has the advantage over the other methods that, through the cutoff parameter, it attempts to select documents that will maximize the utility of the evaluation while minimizing the cost.

8. CONCLUSION

We have presented an algorithm that researchers can use in-house to build test collections incrementally. The algorithm selects documents that are likely to contribute a lot of information about the difference in mean average precision, and stops when it is likely that the difference is meaningful. Our algorithm performs as well as any pooling method and is more likely to generalize to non-TREC collections.

The algorithm uses heuristics that "feel" right. There may be a better weighting scheme, or better stopping conditions. If nothing else, it would be wise to incorporate the inter-topic assumption made by MTF. In the future, we intend to analyze the algorithm more formally to justify or derive its heuristics. We hope that we can quantify the amount of uncertainty in an evaluation with a set of relevance judgments. We'd also like to evaluate our sets of relevance judgments using the *bpref* measure [3] instead of mean average precision, and investigate selecting documents by their effect on bpref.

The algorithm will make it possible for researchers to do in-house evaluations on new corpora and new topics. It is not, however, meant to be a replacement for NIST's processes, where an overriding goal is the creation of a *reusable* test collection that can be adopted with some confidence by a non-participating system. Our approach may result in a broadly useful set of judgments, but it is intented to differentiate between a small set of systems and cannot guarantee more general value.

9. ACKNOWLEDGEMENTS

This work was supported in part by the Center for Intelligent Information Retrieval and in part by SPAWARSYSCEN-SD grant number N66001-02-1-8903. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

10. REFERENCES

[1] S. M. Beitzel, E. C. Jensen, A. Chowdhury, D. Grossman, and O. Frieder. Using manually-build web directories for automatic evaluation of known-item retrieval. In *Proceedings of SIGIR '03*, pages 373–374, 2003.

[2] C. Buckley and E. M. Voorhees. Evaluating Evaluation Measure Stability. In *Proceedings of SIGIR '00*, pages 33–40, 2000.

[3] C. Buckley and E. M. Voorhees. Retrieval evaluation with incomplete information. In *Proceedings of SIGIR '04*, pages 25–32, 2004.

[4] G. V. Cormack, C. R. Palmer, and C. L. Clarke. Efficient Construction of Large Test Collections. In *Proceedings of SIGIR '98*, pages 282–289, 1998.

[5] S. P. Harter. Variations in relevance assessments and the measurement of retrieval effectiveness. *JASIS*, 47(1):37–49, 1996.

[6] M. Kendall. *Rank Correlation Methods*. Griffin, London, UK, fourth edition, 1970.

[7] M. Sanderson and H. Joho. Forming test collections with no system pooling. In *Proceedings of SIGIR '04*, pages 33–40, 2004.

[8] I. Soboroff, C. Nicholas, and P. Cahan. Ranking Retrieval Systems without Relevance Judgments. In *Proceedings of SIGIR '01*, pages 66–73, 2001.

[9] K. Sparck Jones and C. J. van Rijsbergen. Information Retrieval Test Collections. *Journal of Documentation*, 32(1):59–75, 1976.

[10] E. Voorhees. Variations in Relevance Judgments and the Measurement of Retrieval Effectiveness. In *Proceedings of SIGIR '98*, pages 315–323, 1998.

[11] E. M. Voorhees. Evaluation by highly relevant documents. In *Proceedings of SIGIR '01*, pages 74–82, 2001.

[12] E. M. Voorhees. The philosophy of information retrieval evaluation. In *CLEF '01: Revised Papers from the Second Workshop of CLEF*, pages 355–370, London, UK, 2002. Springer-Verlag.

[13] E. M. Voorhees and D. Harman. Overview of the Eighth Text REtrieval Conference (TREC-8). In *Proceedings of TREC-8*, pages 1–24, 1999. NIST Special Publication 500-246.

[14] J. Zobel. How Reliable are the Results of Large-Scale Information Retrieval Experiments? In *Proceedings of SIGIR '98*, pages 307–314, 1998.