Dartmouth College Computer Science Technical Report TR2003-449

# An Active Learning Approach to Efficiently Ranking Retrieval Engines

Lisa A. Torrey
Department of Computer Science
Dartmouth College

Advisor: Javed A. Aslam

## Abstract

Evaluating retrieval systems, such as those submitted to the annual TREC competition, usually requires a large number of documents to be read and judged for relevance to query topics. Test collections are far too big to be exhaustively judged, so only a subset of documents is selected to form the judgment "pool." The selection method that TREC uses produces pools that are still quite large. Research has indicated that it is possible to rank the retrieval systems correctly using substantially smaller pools.

This paper introduces an active learning algorithm whose goal is to reach the correct rankings using the smallest possible number of relevance judgments. It adds one document to the pool at a time, always trying to select the document with the highest information gain. Several variants of this algorithm are described, each with improvements on the one before. Results from experiments are included for comparison with the traditional TREC pooling method. The best version of the algorithm reliably outperforms the traditional method, although its degree of improvement varies.

## 1 Introduction

Every year, more than 100 information retrieval systems are submitted to the Text REtrieval Conference (TREC) ad hoc competition. Each system searches a huge document corpus to retrieve lists of relevant documents for 50 queries. TREC then evaluates all of these lists, which may contain up to 1000 documents each, and ranks the systems based on how well they retrieved the relevant documents for each query.

The method that TREC uses to rank the systems is called *depth-100 pooling*. Since it would be infeasible to read and assign relevance judgments to all of the millions of documents in the corpus, this method was designed to produce a smaller pool to judge. For each query, only the top 100 documents from each list are judged (labelled as either relevant or nonrelevant), and all other documents are assumed to be nonrelevant. Zobel [6] showed that this method is reliable, because by using larger pools he produced only negligible differences from the official results. Rankings created using a pooling depth of 100 can therefore be considered "correct."

Still, even with this reduced number of relevance judgments, the process requires a great deal of human work. In TREC 8, for example, a total of 86,830 documents were judged [5]. An alternative pooling method might significantly reduce the required number of judgments.

### 1.1 Previous Research

Several such alternative methods have been explored. Zobel [6] studied shallower TREC-style pooling and found that pool depths of 50 and above often produced results that were well correlated with the depth-100 rankings. This indicates that it should be possible to find significantly smaller pools that correctly differentiate between systems.

Cormack, Palmer, and Clarke [3] explored methods that attempt to avoid judging nonrelevant documents, thereby decreasing the judgment pool to only the relevant documents. In one experiment, several researchers simply searched the corpus with manual queries until they no longer found relevant documents frequently. In another, they gave priority to documents from lists that were doing well based on previous judgments. They reported good results, although their methods risk being unpredictable and potentially biased.

Soboroff, Nicholas, and Cahan [4] introduced a technique that requires no relevance judgments at all: selecting a pool, determining the expected number of relevant documents, and assigning judgments randomly in proportion to that number. They experimented with pools of varying depth, but always used the top documents in the lists. Their results were surprisingly correlated with the depth-100 rankings, although they tended to rank the best systems badly.

Aslam and Montague [1] investigated the use of metasearch for selecting a pool, using relevance judgments that were either inferred or produced by human assessors. The pools produced by metasearch were substantially smaller than the depth-100 pools, so fewer human judgments were required. The resulting system rankings were well correlated with the depth-100 rankings, although again, the best systems tended to be badly ranked.

These investigations leave open the problem of selecting a truly optimal pool. Assuming that at least some human relevance judgments will be used, the most work will be saved by a technique that can reliably produce optimal pools. Rankings from an optimal pool should be highly correlated with TREC rankings after a minimum number of judgments. The correlation should be especially good among the best systems, which are the most important to score correctly. This paper applies a type of machine learning called *active learning* to this problem.

## 1.2 Active Learning

In the traditional supervised learning framework, the learner is presented with a *hypothesis space* and a pre-prepared set of *training data*. Its task is to narrow down the hypothesis space and select a hypothesis that is consistent with the data. Given enough data, it will rule out everything but the correct hypothesis, which is called the *target concept*. We can view the TREC competition in terms of this framework: the hypothesis space is all possible rankings of the systems, the target concept is the depth-100 ranking, and the training data is the set of human relevance judgments.

Active learning, on the other hand, does not require a static set of training data. Instead, the learner requests one data point at a time. The point that it selects is the one that narrows down the hypothesis space the most, so that it will arrive at the target concept with as few requests as possible. Active learning is an efficient approach when providing data is slow or difficult, because it makes the most out of small amounts of data. We can view the optimal pool problem in these terms: a request for a data point is a request for one human relevance judgment, the target concept is again the depth-100 ranking, and we want to arrive at it using as few relevance judgments as possible.

## 1.3 Information Gain

The amount that labeling a data point would narrow down the hypothesis space is called its *information gain*. In order for active learning to be worthwhile, some data points must have higher information gain than others. In terms of the optimal pool problem, some documents must cause more progress toward the correct ranking than others. Intuitively, this seems to be the case:

- Consider a document that two systems return at the very top of their lists. If it is relevant, they will be rewarded the same amount for retrieving it; if it is nonrelevant, they will be penalized the same amount. In contrast, consider a document that one system returns at the top and the other returns at the bottom. This second document will reward them differently and affect their relative ranks, so it has a higher information gain than the first.

- Consider one document that two systems return in different positions near the top of their lists, and another that they return in different positions near the bottom of their lists. Because documents at the top count more in a system's score than documents at the bottom, the first one has a higher information gain than the second. (The scoring technique is described later.)

- Suppose that several judgments have already been made, and some systems are performing much better than others so far. Consider a document that two good systems disagree on, and another that two bad systems disagree on. For our purposes, the first one has a higher information gain than the second, because we care more about distinguishing the better systems.

### 1.4 The Optimal Pool Problem

This paper reports the results of an active learning algorithm that attempts to produce optimal pools. Five variants of the algorithm were developed as shortcomings were discovered, each building on the last. They are introduced in that order here.

Section 2 explains some standard metrics and graphs used to evaluate the experiments. Sections 3-7 describe the algorithm variants, discussing the issues that came up in each step. They each include results for one query, for purposes of comparison between the methods. These comparative results show both their overall performance against depth-100 pooling and their relative performance at distinguishing the best systems quickly. Section 8 provides a more in-depth look at the best variant, using results from several queries. Section 9 discusses conclusions and future work.

The query used for comparison between the variants is query number 415 from TREC-8, chosen because it displayed fairly well the characteristic progress of each progressive change to the algorithm. The three other queries used to demonstrate the best variant are also from TREC-8, numbers 408, 420, and 445, chosen because they were representative of the range of performance. All queries are from TREC-8 because that was the "hardest" competition to rank in previous research, especially among the best systems.

## 2 Metrics and Graphs

There are four measurements that require brief explanation: one that TREC uses to evaluate systems, and three that we use to evaluate algorithms in comparison to depth-100 pooling. There are also two types of graphs that display information of interest in this paper.

### 2.1 Average Precison

A system's performance on a query is evaluated with a metric called *average precision.* It ranges from 0 to 1, where 0 means the system listed none of the relevant documents, and 1 means the system listed all of the relevant documents before it listed any nonrelevant ones. It is calculated for one system as follows:

- Starting at the top of the list, find the index of each relevant document retrieved.
- At each index, divide the number of relevant documents retrieved so far by the total number of documents retrieved so far, and add this fraction to a cumulating sum.
- Find the total number of relevant documents in the pool (including those not retrieved by this particular system) and divide the sum by this number.

This measure penalizes systems for missing relevant documents, and also for placing nonrelevant documents above relevant ones. Mistakes at the top have more impact, because each document's relevance affects the calculation at every relevant document below it.

### 2.2 Linear Correlation

The metric used to evaluate calculated average precisions is the standard *linear correlation.* Each system is represented by one point: its x-coordinate is the average precision we have calculated, and its y-coordinate is its true average precision. We then calculate the linear correlation of that set of points by finding the line that best fits them and returning a value indicating how well it fits. The range is from -1 to 1, where -1 means they fit perfectly to a line with negative slope, 1 means they fit perfectly to a line with positive slope, and 0 means the values are completely unrelated.

### 2.3 Kendall's Tau

One metric used to evaluate calculated rankings is called *Kendall's $\tau$*. It is a function of the minimum number of pairwise swaps required to turn one ranking into another. It ranges from -1 to 1, where -1 means one ranking is the reverse of the other, 1 means the two are identical, and 0 means they are completely unrelated.

### 2.4 An Alternative Rank Metric

Kendall's $\tau$ is not an ideal evaluation of system rankings, because it gives equal weight to pairwise swaps at both the top and the bottom of the order. Since the ranks at the top are in fact more important, it seems appropriate to introduce a more weighted metric.

An algorithm should be penalized for placing lower-ranked systems above higher-ranked ones, and mistakes at the top should have more impact. These conditions sound familiar, because in fact they are quite similar to the ones explained in the section on average precision. The new metric used in this paper, therefore, is based on the idea of average precision, and will be referred to as *rank precision*. It is calculated as follows:

- Pick an index at random and divide the correct rankings into "good" systems above that index and "bad" systems below it.
- Compute the average precision of the calculated rankings, exactly as if these were relevance judgments and each system was a document.
- Repeat for every possible division index, and take the average of all the resulting values.

This measure can be as high as 1, and it will always be higher than 0. It tends to produce higher values than the Kendall's $\tau$ does, especially once the top systems have been ranked correctly. Both have their highest values at 1, but do not expect a ranking with a very high rank precision to have an equally high Kendall's $\tau$.

### 2.5 Graphs

The first type of graph plots the progress of an algorithm (in terms of one of the above metrics) against the number of relevance judgments made. It will also contain a second curve, which represents the depth-$k$ rankings for $k$=1 through $k$=100, for comparison. In a successful experiment the first curve remains consistently above the second, and it reaches its highest value significantly sooner than the second does.

The second type of graph is a snapshot of the state at one point in time during an experiment. It plots system ranks, which are estimated based on the current set of relevance judgments, against the correct rankings. If the algorithm has reached a completely correct ranking, this graph will be a perfectly straight diagonal line from the lower left to the upper right. In a successful experiment the line forms quickly, especially in the lower left where the best systems are represented.

## 3 Minimizing Uncertainty

After a set of relevance judgments, a system's average precision has a range of possible values. If the rest of the judgments go as badly as possible for that system, its average precision will be at the bottom of that range. If the rest go as well as possible, its average precision will be at the top. During each round of the judgment process, each system has such an *interval* of possible average precisions. Before any judgments have been made, all the intervals begin at 0 and end at 1, but as judgments are added, they shrink. Once all the depth-100 judgments have been made, all the intervals have shrunk to single points and only one value is possible for each system.

The first variant of the learning algorithm, which we will refer to as *A1*, considers the sum of these intervals to be a measure of total uncertainty. During each round of the judgment process, each unjudged document has the potential to decrease that uncertainty by a computable amount. In order to minimize uncertainty as quickly as possible, *A1* selects one document each round for judging, choosing the one whose decrease it expects to be the largest.

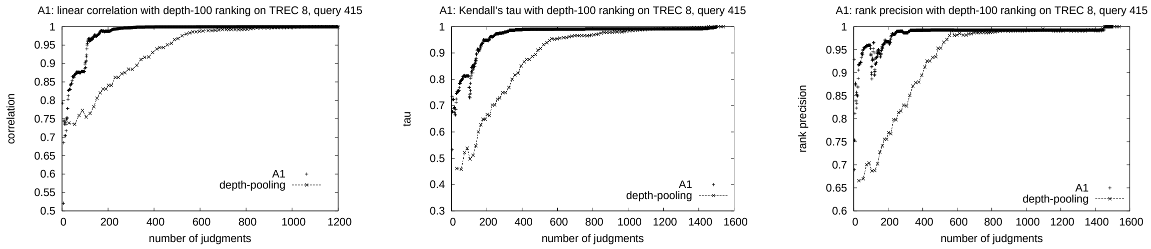

Figure 1: Overall performance of $A1$ and depth-pooling on one TREC-8 query.

## 3.1 Adversarial Choices

There are actually two potential decreases associated with each document: one if it turns out to be relevant, and another if it turns out to be nonrelevant. Before actually asking for the judgment, none of the algorithms know which will be the true decrease. In *A1*, this unknown factor is decided using a minimax strategy; it calculates both values, treats the minimum of the two as the expected decrease, and in the end takes the "maximum of minimums" to choose one document. This is an *adversarial* strategy, and a rather pessimistic one. It ensures that even if relevance judgments are given by an adversary who wants *A1* to make as little progress as possible, decent progress can still be made.

## 3.2 Interval Calculation

To find the minimum and maximum possible average precisions for a system, all unjudged documents must be assigned arbitrary judgments in the worst and best possible ways. For the worst value, any unjudged document that the system does *not* retrieve should clearly be labeled relevant. This label would add to the denominator of its average precision, but contribute nothing to the numerator, thereby decreasing the value. Similarly, for the best value, any unjudged document that the system does not retrieve should be labeled nonrelevant.

Because of the complicated nature of the average precision metric, it is not immediately obvious which will be the "worst" and "best" ways to label unjudged documents *within* a system's list. To find the best value, for example, it seems at first glance that they all should be labeled relevant. But this is not always the case: if the list contains a document that has already been labeled nonrelevant, labeling relevant documents below it may actually lower the average precision.

However, suppose we label all unjudged documents relevant at first. By going from the bottom up and switching them to nonrelevant one at a time, the algorithm will eventually hit upon the best value. That is, given a number of "relevant" labels to assign to documents, it can maximize average precision by clustering them as close to the top as possible. The denominator of the calculation remains constant no matter how the labels are distributed, but the numerator clearly gets larger when the labels are as high in the list as they can be. Similarly, the algorithm will eventually hit upon the worst value by going from the bottom up and labeling unjudged documents relevant one at a time.

The algorithms all calculate minimum and maximum interval values in this way. Since the important documents are those in the depth-100 pool, they assume all other documents are always nonrelevant.

## 3.3 Performance

Given the simplicity of this algorithm, it performed surprisingly well. Figure 1 displays its overall performance on TREC-8 query 415. All of the curves rise well above the depth-pooling curves, indicating that $A1$ reaches the correct average precisions, the correct rankings, and the correct best-system rankings more quickly.

Figure 2 shows the progress of both $A1$ and depth-pooling after approximately 50 judgments, 200 judgments, and 500 judgments. Note that $A1$ produces graphs that are closer to the target, which confirms expectations since its Kendall's $\tau$ is higher at each of these points. Furthermore, while depth-pooling appears to correctly rank the best systems only at the end, $A1$ progresses uniformly, giving equal treatment to systems at the top and bottom of the range.

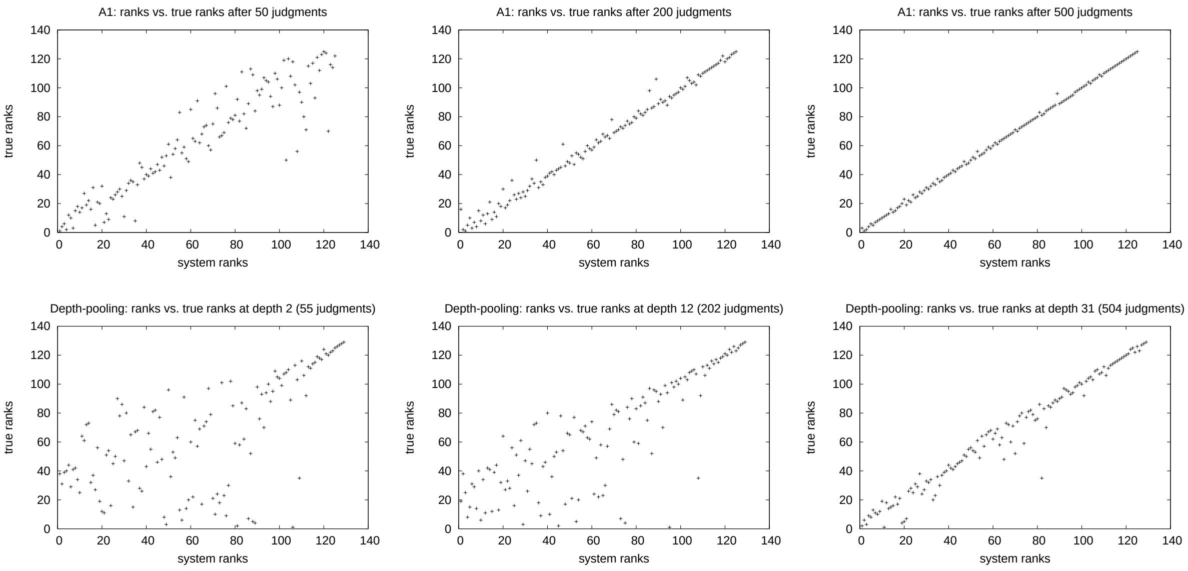

Figure 2: Progress of $A1$ and depth-pooling at three points during the judgment process.

## 3.4 The Next Step

By concentrating on decreasing total uncertainty, $A1$ concentrates on the exact average precisions of the systems, rather than on their relative ranks. While the two are closely related, they are not exactly the same. Consider a system interval that has non-zero size but that does not overlap with any other interval; its rank is certain, even though its average precision is not. But $A1$ may still needlessly spend judgments trying to shrink its size to zero. Since our stated goal is to determine the rankings and not the exact average precisions, the next algorithm should pay direct attention to the rankings.

# 4 Minimizing Overlap

Rather than looking at the total interval distance, $A2$ looks at the total *pairwise overlap* distance. That is, it treats the sum of the system overlaps as a measure of total uncertainty. It measures the length of each system's overlap with every other system, and adds all of these into the total. The document it chooses during each round is the one that should decrease the total the most, according to the same adversarial "minimax" method explained above.

## 4.1 Guessing Precisions

After every round of the judgment process, $A2$ evaluates its progress using linear correlation, Kendall's $\tau$, and rank precision. In order to do so, it needs to calculate average precisions using a partial set of relevance judgments. There seem to be two reasonable ways to use this partial information to guess the values. First: it could assume any unjudged documents are nonrelevant, and calculate average precision as if it had a full set of judgments. Second: it could simply take the average value of each system's current interval, avoiding the potentially dangerous overlap areas as much as possible.

The first of these methods appears to be better in practice. In the long run, the curves are comparable, but the curves produced by the second method tend to have substantial local fluctuation even after the curves produced by the first method have leveled out. This unreliability makes the second method less desirable, since our goal is reliably high performance. There is a likely explanation for the fluctuation: it is usually not possible, using consistent relevance judgments, for every system to fall in the center of its interval. The judgments that would cause one system to do so would make it impossible for others to do so. $A2$, along with all the other variants, therefore uses the first method.

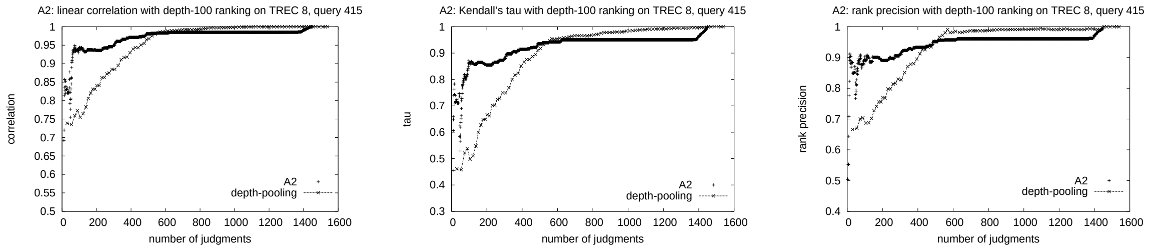

Figure 3: Overall performance of $A2$ and depth-pooling on one TREC-8 query.

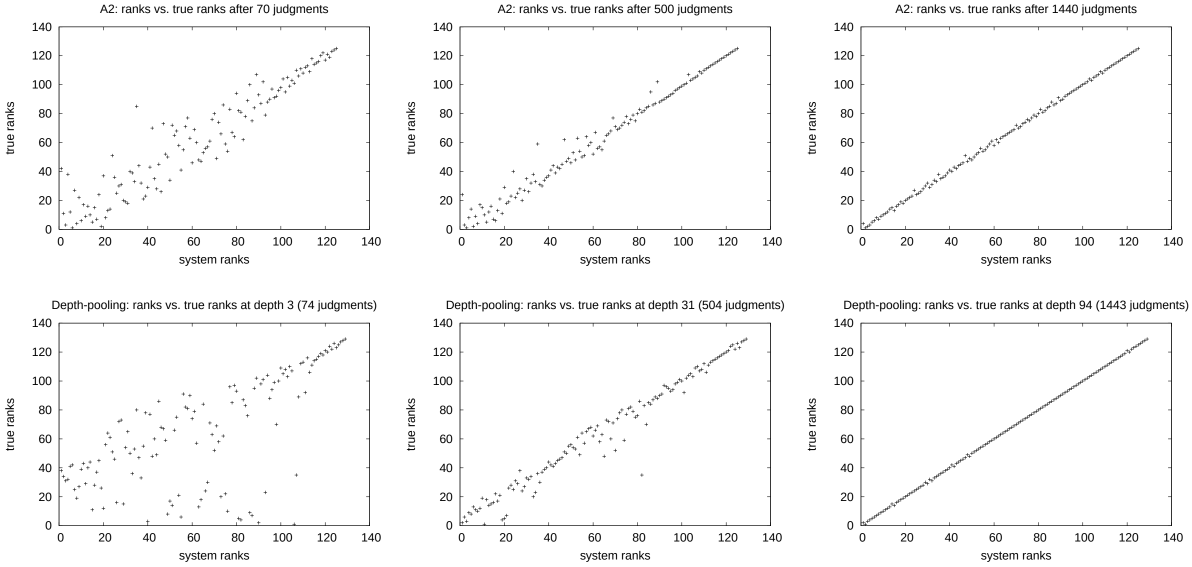

Figure 4: Progress of $A2$ and depth-pooling at three points during the judgment process.

### 4.2 Performance

Figure 3 displays the overall performance of $A2$ on TREC-8 query 415. It actually does worse than $A1$, appearing to level out before reaching the top of the range, and only rising to the top near the end. Figure 4 shows that it also takes a long time to correctly rank the best systems.

We will concentrate on the second issue here, and leave the first for the next section. $A2$ treats all overlaps equally, when in fact some overlaps are more important to eliminate than others. It may be that there are larger overlaps at the bottom of the range than at the top, so that $A2$ first chooses documents that distinguish between the lower systems, only turning to the higher systems at the end.

### 4.3 The Next Step

Overlaps near the top of the range should cost more, since ranking mistakes among the best systems are less acceptable. An algorithm that weights overlaps by their location would be more in accordance with the rank precision measure introduced earlier. Such an algorithm might perform better in terms of rank precision, even if it performs worse in terms of Kendall's $\tau$.

## 5 Weighting Intervals

$A3$ calculates the sum of the pairwise overlaps differently: it multiplies the size of each overlap by its midpoint. This produces a weighted sum that penalizes overlaps in the higher values of the average precision range.

The reason for this particular weighting is that it minimizes the likelihood of overlap among the best systems. The rest of this section is devoted to proving that it does so.

Assume that the systems have been assigned unique integer identifiers. Let the following variables be defined:

$$\begin{aligned} x_{ij} &= \text{the lower boundary of the overlap between systems } i \text{ and } j \\ y_{ij} &= \text{the upper boundary of the overlap between systems } i \text{ and } j \end{aligned}$$

Also let the following functions be defined:

$$d(S_i, S_j) \quad = \quad \begin{cases} y_{ij} - x_{ij} & \text{If } i < j \text{ and systems } i \text{ and } j \text{ overlap at } x \\ 0 & \text{Otherwise} \end{cases}$$

$$m(S_i, S_j) \quad = \quad \begin{cases} \frac{y_{ij} + x_{ij}}{2} & \text{If } i < j \text{ and systems } i \text{ and } j \text{ overlap at } x \\ 0 & \text{Otherwise} \end{cases}$$

The original sum calculation, which simply adds the overlap distances, can then be expressed as:

$$SUM_{orig} \quad = \quad \sum_{i,j} d(S_i, S_j)$$

And the new sum calculation can be expressed as:

$$SUM_{new} \quad = \quad \sum_{i,j} d(S_i, S_j) \cdot m(S_i, S_j)$$

But the original sum calculation can also be expressed in another way, which will allow us to show why the new sum calculation is justified. Let the following functions be defined:

$$f(x) = \text{the number of pairwise overlaps at average precision } x$$

$$v(S_i, S_j, x) \quad = \quad \begin{cases} 1 & \text{if } i < j \text{ and systems } i \text{ and } j \text{ overlap at } x \\ 0 & \text{otherwise.} \end{cases}$$

Then the overlap distance for systems $i$ and $j$ is just $\int v(S_i, S_j, x)dx$. The original sum calculation can then be expressed as:

$$SUM_{orig} \quad = \quad \sum_{i,j} \left[\int v(S_i, S_j, x)dx\right] \quad = \quad \int \left[\sum_{i,j} v(S_i, S_j, x)\right] dx$$

Note that $\sum_{i,j} v(S_i, S_j, x)$ evaluated at one point $x$ is just the number of pairwise overlaps at $x$, which is exactly $f(x)$. So the original sum calculation is also equivalent to:

$$SUM_{orig} \quad = \quad \int f(x)dx$$

The significance of $f(x)$ is that it is proportional to a probability distribution– specifically, the probability that a given overlap encompasses the point $x$. We can find the expected value of a random variable chosen according to that distribution by calculating $\int xf(x)dx$.

This is equivalent to finding the expected average precision at which a randomly selected overlap occurs. This value should be minimized, since that would make overlaps between the best systems very unlikely. We can show that $A3$ does in fact try to minimize it, because the new sum calculation is equivalent to $\int xf(x)dx$. We start by re-writing the integral:

$$\int xf(x)dx \quad = \quad \int x \left[\sum_{i,j} v(S_i, S_j, x)\right] dx \quad = \quad \sum_{i,j} \left[\int x \cdot v(S_i, S_j, x)dx\right]$$

The function $v(S_i, S_j, x)$ has the value 1 between $x_{ij}$ and $y_{ij}$, and the value 0 elsewhere. So we can re-write it again:

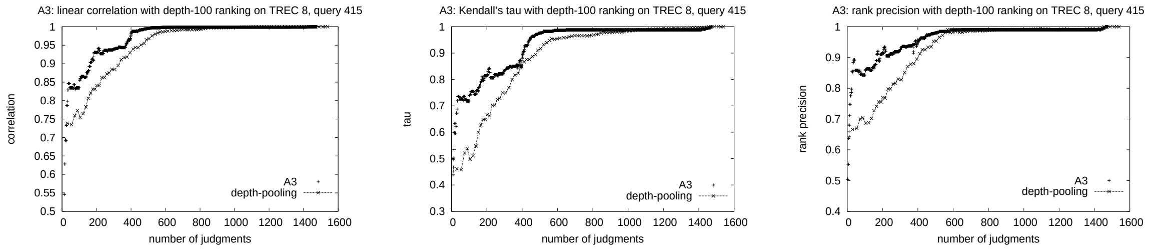

Figure 5: Overall performance of $A3$ and depth-pooling on one TREC-8 query.

$$\int xf(x)dx \quad = \quad \sum_{i,j}\left[\int_{xij}^{y_{ij}} x\cdot 1dx\right]$$

Evaluating the integral, we get the following:

$$\int xf(x)dx \quad = \quad \sum_{i,j}\left[\frac{x^2}{2}\ |_{x_{ij}}^{y_{ij}}\right] \quad = \quad \sum_{i,j}\left[\frac{y_{ij}^2-x_{ij}^2}{2}\right]$$

$$\int xf(x)dx \quad = \quad \sum_{i,j}\left[\frac{(y_{ij}-x_{ij})(y_{ij}+x_{ij})}{2}\right] \quad = \quad \sum_{i,j} d(S_i,S_j)\cdot m(S_i,S_j)$$

This is exactly the new sum calculation: distance multiplied by midpoint for each overlap. So:

$$\int xf(x)dx \quad = \quad SUM_{new}$$

Therefore, when $A3$ tries to minimize the sum, it is always trying to eliminate overlaps among the best systems.

## 5.1 Performance

Figure 5 displays the overall performance of $A3$ on TREC-8 query 415. It improves substantially upon $A2$, especially in the rank precision measure, which indicates that the new sum calculation was successful. Figure 6 confirms this indication, since the best-system rankings appear more quickly than they did in both $A2$ and depth-pooling.

Note that the worst systems still get ranked the most quickly. This is natural, because of the way that these algorithms calculate average precisions with partial judgment sets. All systems start with very low average precisions, which increase gradually as relevant documents are found. The worst systems simply don't have very far to go, so as long as the other systems rise away from them quickly, they will be ranked correctly early on. The important factor to compare is how quickly the best systems rise away from the worst ones.

While the overall performance of $A3$ is better than that of $A2$, it is still below that of $A1$, because the curves rise slowly at the beginning. This seems surprising, since we would expect both $A3$ and $A2$ to improve upon $A1$. Upon further examination, however, there is a likely explanation. Recall that these three algorithms are making adversarial choices: they calculate two possible decreases for each document, assume the smaller is the correct one, and then find the maximum over all documents. In $A1$, the two possible decreases for one document should be nearly equivalent, because an interval can be narrowed equally from the top or from the bottom. $A2$ and $A3$, however, are counting decreases in overlaps. These are much smaller targets than entire intervals, especially at the end of the process. In these algorithms, a document could have one possible decrease that is very high but another that is very low. Since the higher value is ignored, this document would be passed by in favor of one whose possible decreases are both moderate. So although all three algorithms use this conservative approach, it seems likely that $A2$ and $A3$ are held back by it while $A1$ is not.

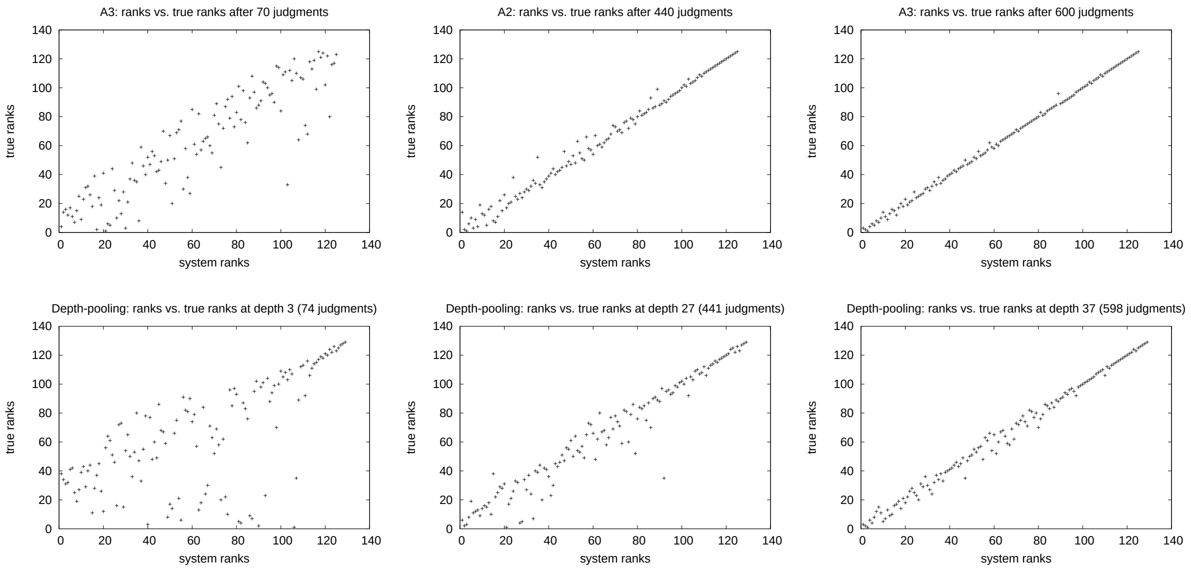

Figure 6: Progress of $A3$ and depth-pooling at three points during the judgment process.

## 5.2 The Next Step

Adversarial choices limit progress, and they are also unnecessarily simple. They assume that every document is equally likely to be relevant or nonrelevant when, in fact, we have more information than that: the lists themselves can be used to estimate the probability that a document is relevant. Using the information inherent in the problem, we could avoid the conservative choices that hold back $A3$.

# 6 Weighting Relevance

$A4$ estimates the probability that each document is relevant, and calculates the weighted average of the two possible decreases. To estimate a probability, it looks at a document's rank in each of the retrieved lists. The lists may not agree, and many of them may overestimate or underestimate the document, but taken all together they can provide a guess.

There is a common curve that characterizes the dropoff of relevant documents in a list [2]. For document $d$ in list $i$ with rank $r_{id}$, its probability $p_i$ of relevance is:

$$p_i(d) \propto \log(r_{id})/r_{id}$$

Constants in this equation will vary with the quality of the list, but they can be ignored here since we are comparing relative values anyway. $A4$ calculates the average $p$ over all lists for each document, and treats this value as the probability that the document is relevant. It then multiplies the "decrease if relevant" by $p$ and the "decrease if nonrelevant" by $(1-p)$, and takes the average.

## 6.1 Performance

Figure 7 displays the overall performance of $A4$ on TREC-8 query 415. The weighted average appears to have improved the algorithm's progress at the beginning of the curve. Figure 8 shows that the best-system rankings are not improved, however. The likely reason is that documents are usually more likely to be nonrelevant than relevant, so the "decrease if nonrelevant" is weighted more. Therefore, $A4$ may be choosing more nonrelevant documents.

Because of the way the algorithms calculate average precisions after each round, relevant labels cause more progress than nonrelevant labels do. By assuming that unjudged documents are nonrelevant, we are

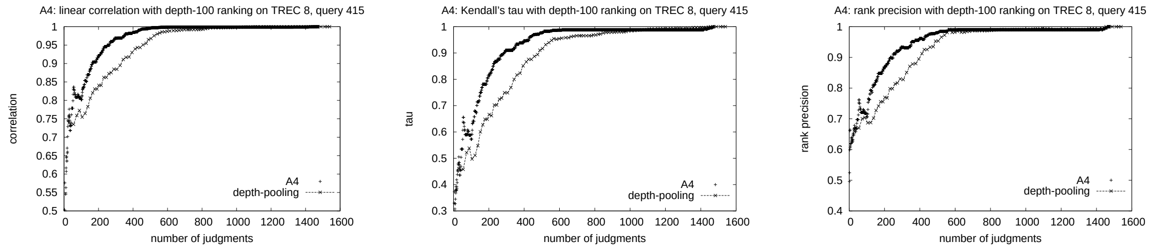

Figure 7: Overall performance of $A4$ and depth-pooling on one TREC-8 query.

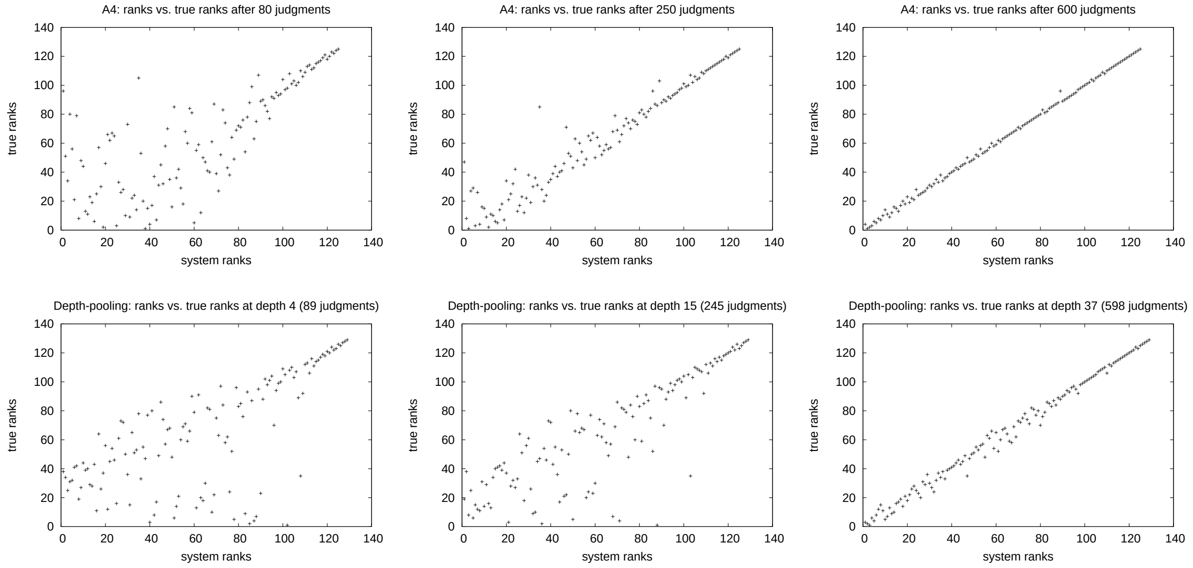

Figure 8: Progress of $A4$ and depth-pooling at three points during the judgment process.

pretending that we know they have nonrelevant labels. In a sense, if we ask for a label and get "nonrelevant," we have actually gained nothing, because we already had that information.

## 6.2 The Next Step

An algorithm might progress more quickly if it used the probability of relevance as a factor when choosing a document. Instead of calculating a weighted average, it could concentrate only on what would happen if the document were relevant.

# 7 Favoring Relevance

$A5$, the final variant, ignores altogether the "decrease if nonrelevant." It calculates only the "decrease if relevant," and weights that value by the probability of relevance.

It is true that $A5$ favors relevant documents. Note, though, that this should not be considered an effective "metasearch" method, because a document with a larger decrease might very well be chosen over one that is more probably relevant.

## 7.1 Performance

Figures 9 and 10 display the performance of $A5$ on TREC-8 query 415. The curves all rise sharply, and the best systems are differentiated quickly. The graph approaches a straight line by 350 judgments out of 1539.

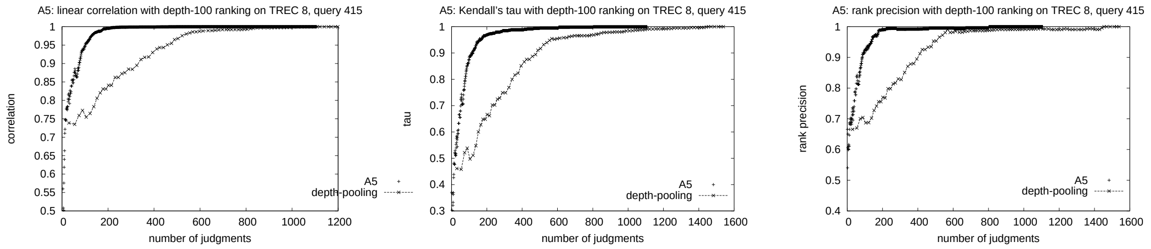

Figure 9: Overall performance of $A5$ and depth-pooling on one TREC-8 query.

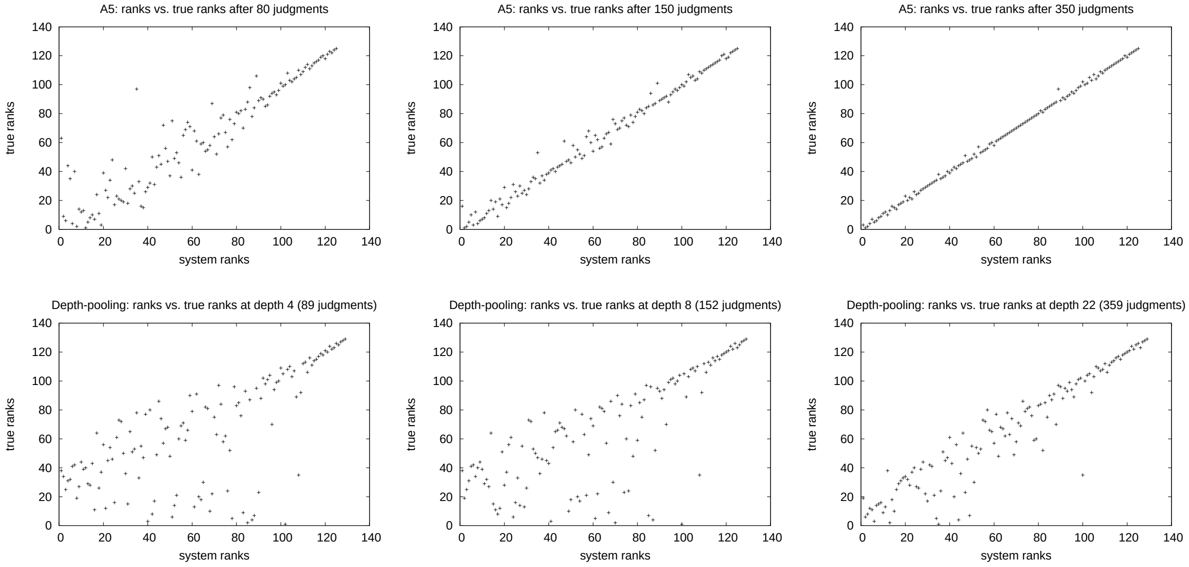

Figure 10: Progress of $A5$ and depth-pooling at three points during the judgment process.

# 8 Results

The remaining figures display results of three more queries from TREC-8 (408, 420, and 445). They demonstrate the range of performance of $A5$. In all queries it performs better than depth-pooling, but sometimes not by a great amount.

Query 408 (Figures 11 and 12) is an example of how the algorithm can produce better rankings than average precisions– while its Kendall's $\tau$ is higher than that of depth-pooling, its linear correlation is lower. Its rank precision is sometimes better, sometimes worse, overtaking the depth-pooling rank precision in the end. This query seemed to be a difficult one, because both methods struggled to rise to the top of the range.

Query 420 (Figures 13 and 14) is an example of how the algorithm can score better on rank precision than on Kendall's $\tau$. While it falls behind in the latter graph, it is consistently ahead in the former. Its linear correlation is also consistently ahead of the depth-pooling curve. This query seemed to be a particularly easy one, because both methods rose to the top of the range very quickly.

Query 445 (Figures 15 and 16) is another example of a good performance by $A5$. As with query 415, the curves rise sharply and remain above the depth-pooling curves. It seems that queries of average difficulty, such as 415 and 445, are the ones on which this algorithm performs best.

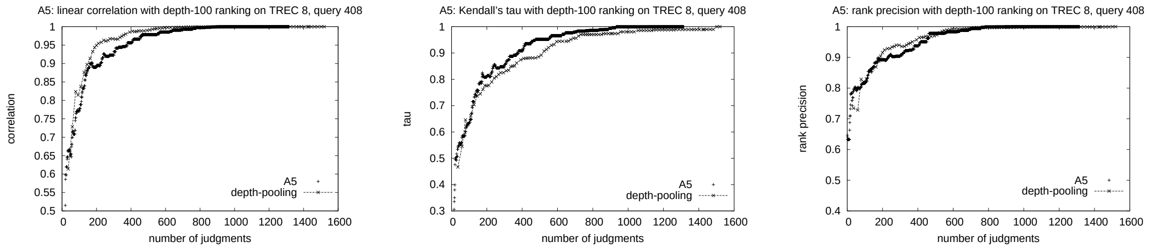

Figure 11: Overall performance of $A5$ and depth-pooling on query 408.

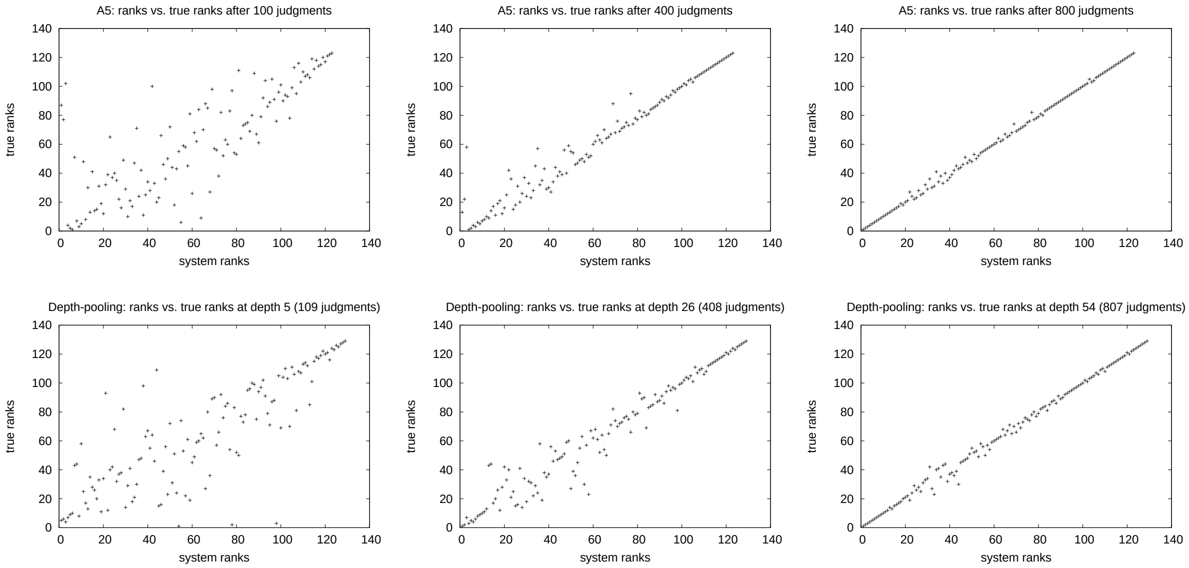

Figure 12: Progress of $A5$ and depth-pooling at three points during the judgment process.

# 9 Conclusions

The algorithms introduced in this paper compare favorably to the depth-pooling method on queries taken from the “hardest” TREC competition thus far. The early variants were often, although not reliably, faster than depth-pooling at both ranking systems and distinguishing the best systems. The final variants have been reliably better in all experiments so far, although the degree of improvement varies. The result is the beginning of an answer to the optimal pool problem.

There are more issues to address in future experiments. The first should be algorithm efficiency, which was a severely limiting factor here. Since our goal was to be reliable, we evaluated the effects of every single unjudged document in each round of the algorithms. This took stunning amounts of time. Despite their performance, therefore, the algorithms are not currently very practical. It may be possible to avoid considering every unjudged document before choosing one; the challenge would be to do so without sacrificing reliability.

If the algorithms could be made significantly faster, the next issue would be to conduct more wide-scale experiments than were possible here, examining their performance over entire TREC conferences instead of individual queries. Then further improvements could be made to the strategies, especially in terms of distinguishing the best systems. It might also be interesting to find the actual optimal pools, by “peeking” at the relevance judgments before choosing documents, for comparison to the results of these algorithms.

In the end, this method may be inherently too expensive or unpredictable to be useful in a setting like TREC, where there is much concern about fairness. However, it is also possible that a truly optimal pool

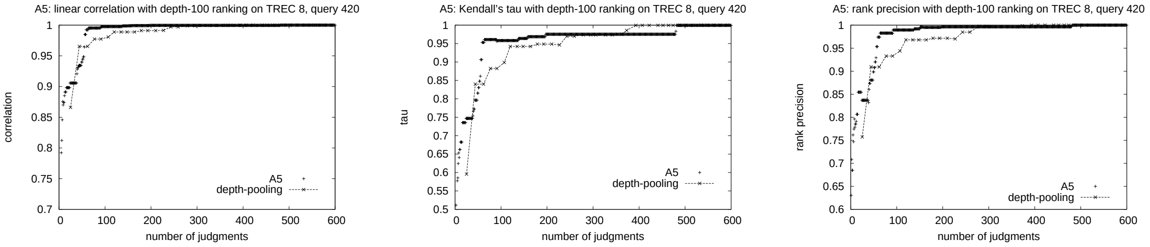

Figure 13: Overall performance of $A5$ and depth-pooling on query 420.

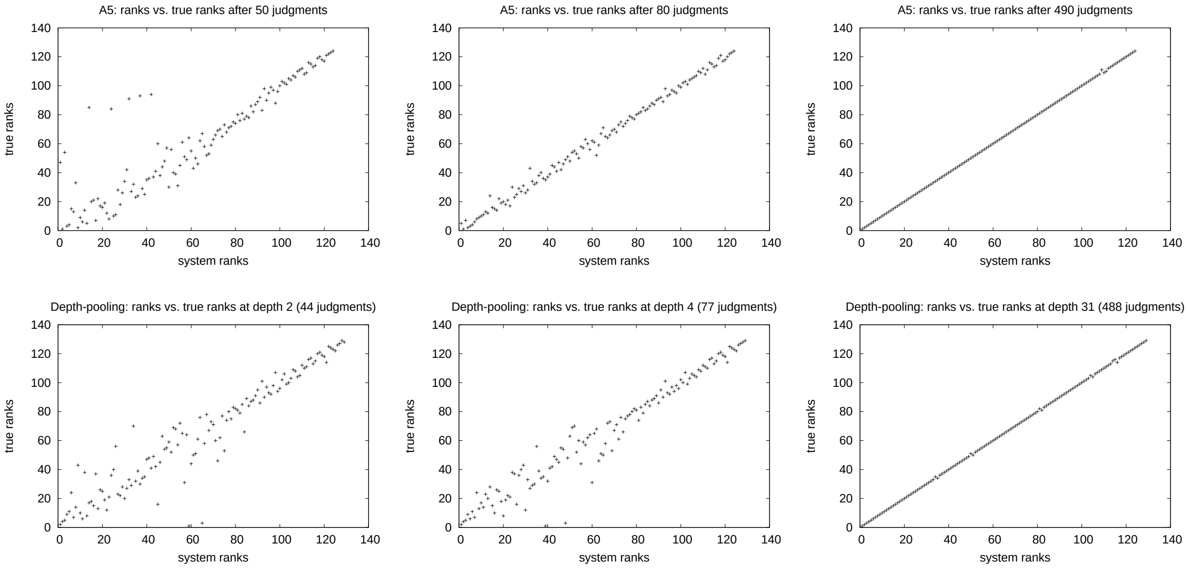

Figure 14: Progress of $A5$ and depth-pooling at three points during the judgment process.

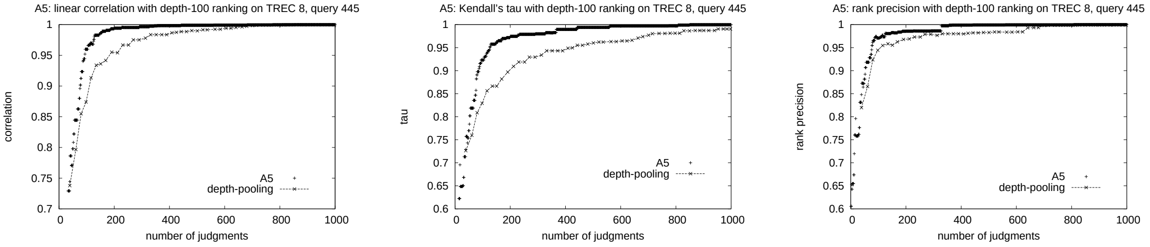

Figure 15: Overall performance of $A5$ and depth-pooling on query 445.

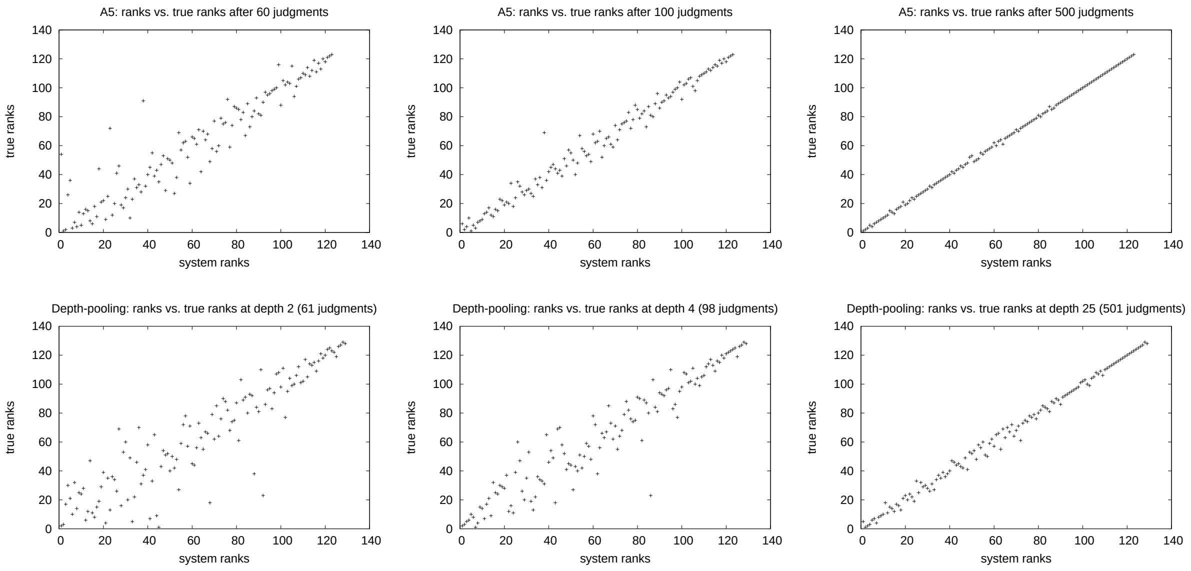

Figure 16: Progress of $A5$ and depth-pooling at three points during the judgment process.

can be calculated in a reasonable time, in which case the method would be entirely predictable. If that were the case, it would allow TREC to continue to grow at its current rate and yet remain manageable for some time to come.

# References

[1] J. Aslam and M. Montague. Evaluating Retrieval Systems with Few or No Relevance Judgments via Metasearch. Submitted to *Sigir '02*.

[2] J. Aslam. Personal communication, April 2003.

[3] G. Cormack, C. Palmer, and C. Clarke. Efficient construction of large test collections. In Croft et. al. [5], pages 282-289.

[4] I. Soboroff, C. Nicholas, and P. Cahan. Ranking retrieval systems without relevance judgments. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* [1], pages 66-73.

[5] E. Voorhees and D. Harman. Overview of the Eighth Text REtrieval Conference (TREC-8). In D. Harman, editor, *The Eighth Text REtrieval Conference (TREC-8)*, Gaithersburg, MD, USA, 2000. U.S. Government Printing Office, Washington D.C.

[6] J. Zobel. How reliable are the results of large-scale retrieval experiments? In Croft et al. [5], pages 307-314.