Constructing Test Collections by Inferring Document Relevance via Extracted Relevant Information

Shahzad Rajput Matthew Ekstrand-Abueg Virgil Pavlu Javed A. Aslam
College of Computer and Information Science, Northeastern University
{rajput, mattea, vip, jaa}@ccs.neu.edu

ABSTRACT

The goal of a typical information retrieval system is to satisfy a user's information need—e.g., by providing an answer or information "nugget"—while the actual search space of a typical information retrieval system consists of documents—i.e., collections of nuggets. In this paper, we characterize this relationship between nuggets and documents and discuss applications to system evaluation.

In particular, for the problem of test collection construction for IR system evaluation, we demonstrate a highly efficient algorithm for simultaneously obtaining both relevant *documents* and relevant *information*. Our technique exploits the mutually reinforcing relationship between relevant documents and relevant information, yielding document-based test collections whose efficiency and efficacy exceed those of typical Cranfield-style test collections, while also generating sets of highly relevant information.

Categories and Subject Descriptors: H.3.3 [Information Search and Retrieval]

General Terms: Performance measurement

Keywords: information retrieval, relevance assessment, evaluation, nuggets

1. INTRODUCTION

Retrieval systems are traditionally evaluated by (1) constructing a test collection of documents (the "corpus"), (2) constructing a test collection of queries (the "topics"), (3) judging the relevance of the documents to each query (the "relevance judgments", or qrel file), and (4) assessing the quality of the ranked lists of documents returned by each retrieval system for each topic using standard measures of performance such as precision-at-cutoff, nDCG, average precision, and so forth.

Much thought and research has been devoted to each of these steps in, for example, the annual text retrieval conference TREC [19]. While most of TREC tasks use *documents as units of relevance*, in recent papers [17, 15, 14, 28, 31] it has been shown that many evaluation tasks such as novelty, diversity, ambiguity, redundancy and entailment require a finer granularity. This can be obtained either directly by assessing fragments or *nuggets* of text [15, 28, 33], or indirectly by deciding if any two assessed documents are on the same "subtopic" [14]. Clearly the former is more powerful, but also far more expensive, as it requires that the assessors produce relevance grades for perhaps millions of nuggets extracted from relevant documents. To our knowledge, no previous work that uses some form of nuggets as the unit of relevance presents an efficient and reliable method to obtain such nuggets; such an efficient and reliable methodology is the subject of this paper.

In previous work [28], we demonstrated the power of a carefully constructed set of nuggets for building test collections. Briefly, for each query, a sample of documents is judged by an assessor, and from the judged relevant documents, the assessor extracts relevant nuggets of information in the form of sentences.[1] This set of relevant nuggets is then used to *infer* the relevance of all unjudged documents via a form of text matching: Those documents containing such known and extracted relevant information are (automatically) judged "relevant", while those not containing relevant information are judged "non-relevant". The resulting judged pool—containing both inferred and actual relevance assessments—is far more comprehensive than typical Cranfield-style pools, and the quality of such pools is demonstrated by the accuracy of both the inferred assessments and, in turn, their assessments of retrieval systems (both those that contributed to the pool and those that did not).

While nugget-based assessments clearly represent important progress over the typical Cranfield-style pooling technique, some aspects of the existing nuggets-based methodology are impractical, principally:

1. The manual extraction and processing of nuggets is time consuming.

2. The sample of documents to be judged is fixed and drawn in advance; i.e., it is a "batch" process that cannot adapt "on-line", given a partial set of judged documents and nuggets.

In this paper we present a methodology that solves these

[1] In our past experiments, approximately 200 documents per query were judged, yielding approximately 87 (relevant) nuggets extracted per query, on average. The limited number of relevant nuggets found was primarily due to the difficulty of manual nugget extraction, a problem we directly address in this work.

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.

problems by adapting the well known principle of "Active Learning" [35, 7]: If the learning algorithm can intelligently *choose* the instances on which to train, then it can iteratively select the most useful data points, obtaining the same learning performance with only a small fraction of the training set.

To apply Active Learning to the assessment process, we first recognize that there is a natural *mutual relevance reinforcement* between nuggets and documents: good nuggets match relevant documents and relevant documents yield good nuggets. As such, we devise an interactive active learning process that (1) maintains sets of judged documents and candidate relevant nuggets, (2) uses the set of nuggets to select new documents for assessment, and (3) uses the set of judged documents to extract more candidate relevant nuggets. Such a framework shares commonalities with Reinforcement Learning [39] and the Expectation-Maximization (EM) algorithm [9].

Our main contribution is handling the *extraction* and *scoring* of candidate relevant nuggets completely automatically; neither require direct input from the assessor, thus minimizing the effort to 30% or less of the typical Cranfield-style judging effort. (Assessors provide *indirect* input on nuggets by judging documents.) Our nuggets-based qrel contains both *actual* and *inferred* relevance assessments, and it is comprehensive in the sense that any document can be automatically assessed for relevance given the set of relevant information extracted in the form of nuggets. The actual and inferred document relevance assessments together form our "nugget-based qrel", which can be used to accurately and efficiently evaluate IR systems using standard performance metrics. Furthermore, given that our qrel is comprehensive via inferred relevance assessments, we demonstrate that our nuggets-based evaluations are quite *reusable*, accurately evaluating even unseen (and unpooled) systems that retrieve large number of unique relevant documents. In effect, our methodology evaluates systems based on whether they accurately and comprehensively retrieve *relevant information* rather than merely whether they retrieve a limited and incomplete set of judged *relevant documents*.

Furthermore, though not explored in this work, the set of relevant nuggets may themselves be of independent interest as this set corresponds to the collected *relevant information* for a given topic. One can imagine, for instance, using such nuggets to evaluate topical summaries, to obtain clusters in information-space corresponding to natural subtopics, and so forth. These and other future directions are discussed in Section 6.

We begin by discussing related work in Section 1.1. Section 2 describes the technical details of our proposed nuggets-based methodology. Section 3 details experiments using three TREC datasets: Adhoc99, Web09 and Robust05. In Section 4, we provide an analysis of the quality of our automatically extracted candidate relevant nuggets. Section 5 gives a detailed analysis of those cases where our methodology succeeds and those where it does not (together with potential remedies), and we conclude with future work in Section 6.

1.1 Related Work

Judging documents for relevance cannot scale with modern collections. TREC, a U.S. government effort, coordinates professional assessors to judge up to 3,000 documents per query (ad hoc tracks). While this is a significant effort, it has been shown that there are large assessor disagreements, fatigue driven labeling mistakes [13], and a large number of relevant documents not even considered for assessment as the number of relevant documents can reach millions [28]. Several methods have been proposed to address the problem of scale [5, 12], and while they are efficient at sampling the document space, they are still dependent on the number of documents considered, which can grow into the billions. However, the actual amount of information/nuggets relevant to the query remains small.

Several specific evaluation needs, including diversity measurements, require relevance grades on a sub-document scale. These methods often rely on subtopic assessment by obtaining more than one judgment for each document [14]. Not only are these subtopics sometimes poorly imposed, but the assessment effort remains directly proportional to both number of subtopics and the number of documents.

Researchers have shown the benefits of nuggets [23], but have not provided an easy way to obtain them. Crowdsourcing [2] is faster and cheaper at assessing documents; however, it is not more scalable than TREC methods and is far less reliable [22]. Nuggets of a different nature are widely used in the evaluation of question answering systems [17, 42], where the quality of the system is determined by having humans assess if the output contains correct answers. [26, 27] propose a method to automatically evaluate the quality of a question answering system's output by matching it with manually extracted correct answers. However, their method is not likely to work when the output of the system is of the size of a document. Similarly, different forms of nuggets are also used in a limited context for novelty and diversity [15]. In the former case, the nuggets tend to be very short and specific answers to "who", "when", and "where" questions. In the latter case, nuggets were not used as the sole basis of evaluation and vast quantities of document judgments were still required. Others have proposed using sets of relevant and non-relevant keywords to assess arbitrary documents [3], though we believe that nuggets can capture more information than simple keywords. Another issue with this approach is the use of non-relevant keywords: In principle, a document can contain any amount of non-relevant information; it is relevant if and only if it contains at least one piece of relevant information. [30] uses nuggets to evaluate the quality of a system's response to a given query; however, the nugget extraction and the evaluations are performed entirely manually. Our previous work [28] tackles the problem in the general domain using manual nugget extraction as well.

The use of click-through data has also been proposed [29], but this is only applicable to the web and only for those queries and documents with sufficient "clicks." Evaluating IR systems without relevance assessments has also been the subject of research [36, 38, 32]; however, these methods tend to lack the capability of inferring the relevance of arbitrary documents outside the pool, limiting reusability.

Finally, Intensive Natural Language Processing has been used in recognizing textual entailment [8], which is a significant part of our problem too. While we are not employing sophisticated NLP methods in this paper, we believe that an improved matching can be obtained by embedding a minimum of NLP to detect synonyms and negations.

2. METHODOLOGY

The fundamental thesis of this paper is that finding relevant information immediately leads to relevant documents, which in turn lead to more relevant information. Additionally, marked non-relevant documents can determine non-relevant information; in particular, TREC assessment philosophy that a document is relevant if it contains any bit of relevant information fits our thesis well: any nuggets found in marked non-relevant documents *must be non-relevant*, up to reasonable assessor disagreement and label noise.

2.1 Concept

We are proposing a framework of mutual, iterative reinforcement between nuggets and documents, most similar to Active Learning [35, 7] mechanisms. As in Active Learning, the feedback (relevance assessment) is given *iteratively* only on the documents/data. Thus, for our IR setting, no effort is added to the current assessor practice.

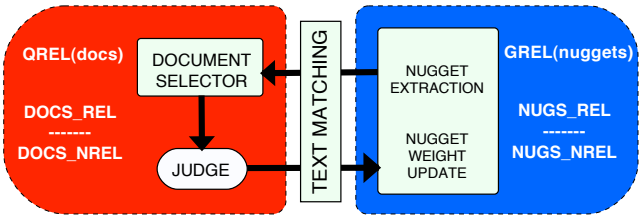

Figure 1: The overall design of assessment process: Iteratively, documents are selected and assessed, and nuggets are extracted and [re]weighted.

Figure 1 illustrates such framework, with the MATCHING being the reinforcing procedure from documents to nuggets and vice-versa:

- the iterative loop selects documents based on the current notion of relevance (as expressed by current set of nuggets G); documents are judged and put into the set of documents (docs QREL);
- for relevant documents, all nuggets are extracted and added to G
- all nugget weights/qualities are updated given the relevant and non-relevant documents judged so far and how well they match each nugget

This framework uses three components that can be designed somewhat independently. (1) The text MATCHING between nuggets and documents is both the syntactic problem of measuring whether the nugget and the document describe the same people, places, locations, dates, facts, actions, etc and the semantic, or textual entailment problem of reflecting the fact that the document entails (presents the semantical information found in) the nugget. (2) The document SELECTOR decides what documents to be judged next, given the current nuggets and their weights, matches, and possibly other contextual information like position of documents in retrieved lists, or the specifics of task, or the desired balance between exploration and exploitation. Finally, (3) the nugget EXTRACTION is responsible for deciding what nuggets to consider from judged-relevant documents, and also responsible in deciding how much each nugget currently matters (updating nugget weight).

A fourth component, not explored in this paper, is the JUDGE model. Here we assume the judge follows the traditional NIST assessor, simply labeling binary documents as relevant (contains something of some relevance) or non-relevant; a more sophisticated judge can review and modify the nuggets extracted, categorize documents, use graded relevance, annotate important keywords, or classify the query.

2.2 Implementation

We present our chosen implementations for SELECTION, EXTRACTION and MATCHING, noting that for each of these we settled on these methods after trying various ideas. While ours work well, each can be independently replaced by more suited techniques for specific tasks.

Text MATCHING. As presented in [28], the matching algorithm is based on a variant of *shingle matching*, which is often used in near-duplicate detection [10, 11]. A shingle is a sequence of k consecutive words in a piece of text. For example, after stopwording, the nugget `"John Kennedy was elected president in 1960"` has the following shingles for $k = 3$: (`John Kennedy elected`), (`Kennedy elected president`), and (`elected president 1960`).

Shingle score: For any nugget and each shingle of size k, let S be the minimum *span* of words in the document that contains all shingle words in any order. A shingle matches well if it is contained in a small span. We use the algorithm presented in [25] to find the shortest span of shingle words in a text document in linear time. We define the shingle matching as

$$shingleMatch = \lambda^{(S-k)/k}.$$

where λ is a fixed decay parameter. We found $\lambda = 0.95$ to be an effective value by trial and error. A shingle that matches "perfectly" will have a score of 1. Note that, in contrast to standard *shingle matching* used for duplicate detection, we do not require all shingle words to be present in the matching document in the same order or contiguously. Our method is inspired by near-duplicate detection, but is in fact quite different. High scores indicate a match of known relevant information, not necessarily of redundant or duplicate text. Furthermore, while a known nugget is required to be present in a document for a good match, the document often contains new/unknown relevant information as well.

Nugget matching: To obtain a matching score for nugget n against document d, we average the scores for each of nugget shingles:

$$\begin{aligned} nuggetmatch &= M(n,d) = \\ &= \frac{1}{\#shingles} \sum_{s \in shingles(n)} shingleMatch(s,d) \end{aligned}$$

While looking for a good but simple text matching strategy, we considered two bag-of-words possibilities: BM25 and Jaccard's Coefficient; both significantly underperformed the presented shingle technique (see 3.1). We speculate that more sophisticated NLP techniques can be used to improve the matching, possibly inspired from textual entailment [8].

Document SELECTION. At each iteration of the document - nugget loop, one or more documents are selected and immediately assessed by the judge. The selection has two parts: First, a ranking function ranks all candidate documents based on a score obtained by matching existing nuggets: at round r, G is the current set of nuggets $n \in G$,

each with current quality weight q_n. Then the document score is a dot product of quality and matching

$$DocScore(d) = \sum_{n \in G} q_n * M(n, d)$$

Since some documents can equally match the nugget set (a zero match in the worst case), we add to the document score a retrieval value considering the positions of the documents in various retrieved lists (submitted IR systems in TREC case). Such methods have been developed before[6, 16, 12]; we use the method described in [6].

Second, a probabilistic procedure picks a number of documents from the ranking. We found that a good balance between exploitation and exploration can be achieved by sampling from a scaled geometric distribution

$$\alpha(K)p(1-p)^{r-1}$$

associated with the document ranks r (not scores), with the geometric base $p = 0.4$ found empirically.

Nugget EXTRACTION and Weighting. Nuggets are extracted *only from relevant documents* by a simple NLP script which looks for sentences; stemming and stopwording is performed on the nuggets to improve matching specificity. We treat the nuggets as *experts* and at each iteration we compute the *current nugget quality* q_n, an *importance weight* used by the document selector when computing document scores. These weights are only an indication of the current belief that a nugget is relevant: if matching documents are proven non-relevant, the weight becomes insignificant; a nugget only maintains a high weight across many iterations if it consistently matches relevant documents.

The weighting mechanism is adapted from experts combination/online learning, where the problem has been exhaustively studied. We use the "Hedge" algorithm[18], the internal engine of AdaBoost and Rankboost (but without boosting), which has been shown to have good theoretical properties [34] and fast implementations. A relevant document d increases the weight of the matching nugget n based on the matching score:

$$q_n^{new} = q_n^{old} / \beta_1^{M(n,d)}, normalized$$

while non-relevant document decreases the weight:

$$q_n^{new} = q_n^{old} * \beta_2^{M(n,d)}, normalized$$

Some nuggets are extracted very early in the loop, while others enter G very late; for consistency, the initial weight of each nugget is computed as if the nugget existed in G all along. The two beta values are found by trial and error and for our systems are set at $\beta_1 = 0.8$ and $\beta_2 = 0.5$. This hedge variant with β values associated with each feedback outcome (relevant, non-relevant) was first described in [4]. Other minor heuristics are used to prevent nugget weights from being extremely large. A particularly important feature of Hedge is its adaptability[18]: nugget quality is high as long the nugget brings in relevant documents; after that, it decreases if non-relevant documents match the nugget, or the nugget becomes irrelevant (irrespective of quality q_n) if no other documents match it.

3. EXPERIMENTS

In this section, we describe experiments to show the ability of our process to quickly form test collections by achieving a high recall with high precision. Additionally, we demonstrate how creating a set of nuggets with weights allows us to infer relevance of documents in systems not present during the initial judgments. Finally, we show justification for our choices of the components and how they can be used to improve other procedures. All of this is done using far less manual effort with little reduction in utility as compared to our previous work.

We evaluate the performance of our system against three datasets of varying complexity and document types: AdHoc99, Robust05, and Web09. The specifics of these datasets can be seen in Table 1.

Dataset	Collection	Queries	Docs Assessed	Relevant Docs
AdHoc99	Tipster disks	50	86,830	4,728
Robust05	AQUAINT	50	37,798	6,561
Web09	Clueweb09	50	40,642	10,313

Table 1: Queries and Documents per collection.

The Ad Hoc track 1999 [1] uses a collection of half a million clean text newswire articles. This is also referred to as the ad hoc track TREC 8. With an average of 1,736 documents assessed (depth-100 pool) for each of 50 queries, it is considered to have effectively complete relevance assessments. There were 129 IR systems submitted to this track.

The Robust track 2005 [41] is a more challenging dataset that uses one million clean text documents from the AQUAINT collection. It has an average of 756 documents assessed for each of 50 queries. The queries used in this track are known to be difficult. There were 75 IR systems submitted to this track.

The Web track 2009 [14] uses the ClueWeb09 collection of one billion web documents. With an average of 812 documents assessed (depth-10 pool) for each of 50 queries, it is considered to have incomplete relevance assessments. There were 120 IR systems submitted to this track. The web track possess additional challenges: (1) some of the documents are difficult to parse due to html complexities, (2) detecting spam on the web is a well known but not solved problem, and (3) transforming the web documents to clean text is a challenge; in particular, removing menus and ads from the documents is non-trivial.

Due to issues in basic sentence extraction, text was not extracted from all documents. For our methodology, this is equivalent to finding no nuggets in a document, and therefore using only the ranking score during document selection.

3.1 Recall

We analyze how quickly we obtain effectively complete document relevance assessments. We also compare our results against depth-pooling, the current standard for TREC collection generation, as well as relevance feedback and Support Vector Machines (SVM) ranking.

In this paper, we used SVMRank [24] for ranking unassessed documents. For Relevance Feedback [37], we used top 20 terms from labeled relevant documents to modify the query and rerun BM25. The process is repeated a number times.

Figure 2 shows that the Nugget-based Test Collections consistently require fewer documents to achieve all recall values as compared to depth-pooling, relevance feedback and support vector machines. For instance, obtaining 90% recall on AdHoc99 requires the Nugget-based method, depth-

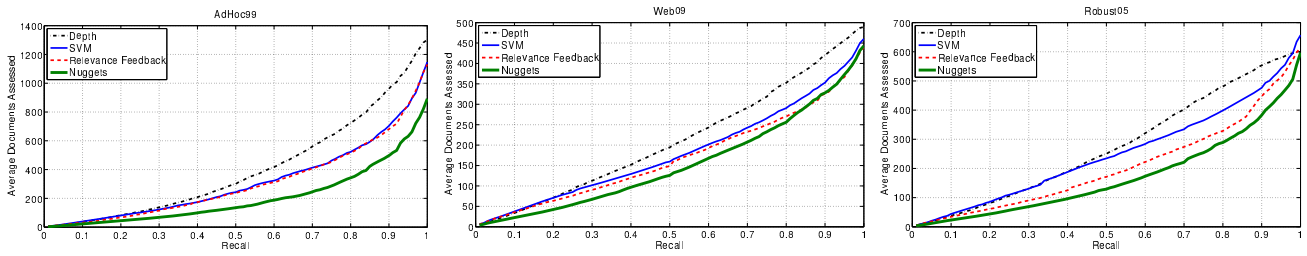

Figure 2: Recall vs. Average Document Assessed

	Recall				
	60%	70%	80%	90%	100%
AdHoc99					
Depth-pooling	417	561	724	962	1,302
Relevance Feedback	313	407	516	681	1,130
SVM	323	413	523	704	1,147
Nuggets	187	245	343	498	892
Web09					
Depth-pooling	248	291	352	418	487
Relevance Feedback	196	232	263	326	447
SVM	201	243	291	353	460
Nuggets	129	169	245	325	448
Robust05					
Depth-pooling	321	402	481	553	600
Relevance Feedback	222	274	328	447	614
SVM	284	334	398	477	657
Nuggets	183	231	287	378	594

Table 2: Number of documents assessed: The nuggets method requires less assessed documents than other methods to achieve the same recall.

pooling, relevance feedback and support vector machines to assess 498 documents, 962 document, 681 documents and 704 documents respectively. In this case, the Nugget-based method achieves an improvement of about 48% over depth-pooling, 27% over relevance feedback and 29% over support vector machines. Similarly, obtaining 100% recall on AdHoc99 requires the Nugget-based method, depth-pooling, relevance feedback and support vector machines to assess 892 documents, 1,302 document, 1,130 documents and 1,147 documents respectively. In this case, the Nugget-based method achieves an improvement of about 31% over depth-pooling, 21% over relevance feedback and 22% over support vector machines. See Table 2 for comparisons for various different recall points on AdHoc99, Web09 and Robust05.

Shingle-matching vs simple BM25. We evaluate to a small extent the validity of our choice of matching nuggets with documents, were we could have used a standard IR function like BM25. For this comparison, we match the nuggets with documents using BM25 instead of using shingle-matching, and all other components are kept constant. Table 3 provides recall comparisons. It is evident from this table that nuggets matching with documents using the shingle-based method obtains a significantly higher recall as compared to the recall obtained when using BM25.

3.2 IR System Evaluation

The Nugget-based method iteratively selects documents

	Average Documents Assessed					
Method	50	100	150	200	250	300
AdHoc99						
Shingles	0.3934	0.5351	0.6262	0.6898	0.7370	0.7785
BM25	0.3987	0.5077	0.5877	0.6553	0.7046	0.7470
Web09						
Shingles	0.1916	0.3142	0.4125	0.4973	0.5660	0.6179
BM25	0.1683	0.2646	0.3450	0.4089	0.4761	0.5406
Robust05						
Shingles	0.2849	0.4672	0.5834	0.6631	0.7349	0.7893
BM25	0.2402	0.4055	0.5191	0.6177	0.6795	0.7369

Table 3: Recall comparison when nuggets are matched with documents using Shingles vs BM25.

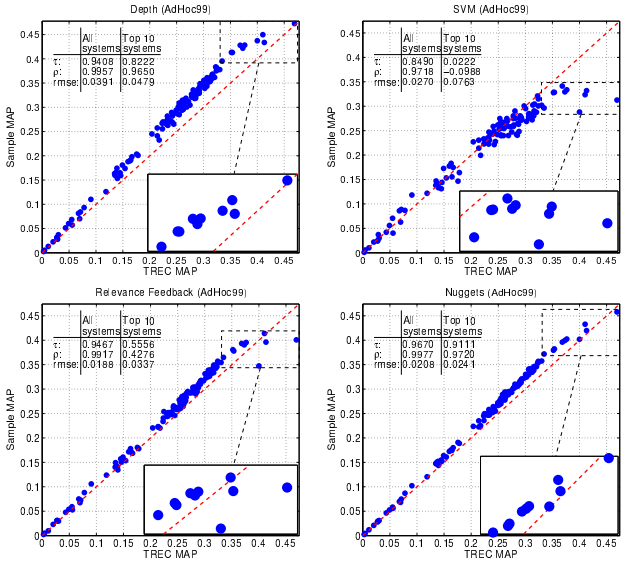

Figure 3: IR System evaluation performed by assessing 350 documents on AdHoc99.

to be assessed, assesses the selected document, and extracts and [re]weights the nuggets.

The assessor may decide to stop at any point. The set of documents assessed together with the assessments forms the nugget-based qrel at that point. This qrel, also referred as the "Sample," can then be used to evaluate the systems.

We produce four separate qrels (each by assessing 350 documents for all queries) based on: depth-pooling, relevance feedback, support vector machines, and the Nugget-based method. Using these qrels, we evaluate all systems over all 50 queries separately for AdHoc99, Web09 and Robust05. The results for AdHoc99, Web09 and Robust05 are shown in Figures 3, 4 and 5 respectively, with each data point

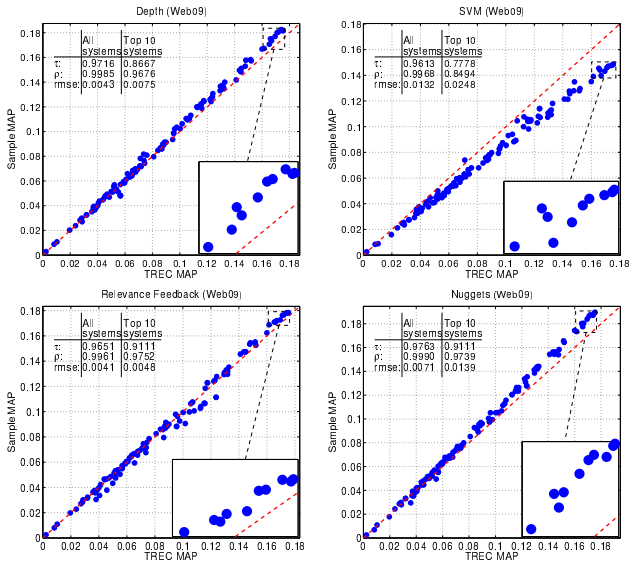

Figure 4: IR System evaluation performed by assessing 350 documents on Web09.

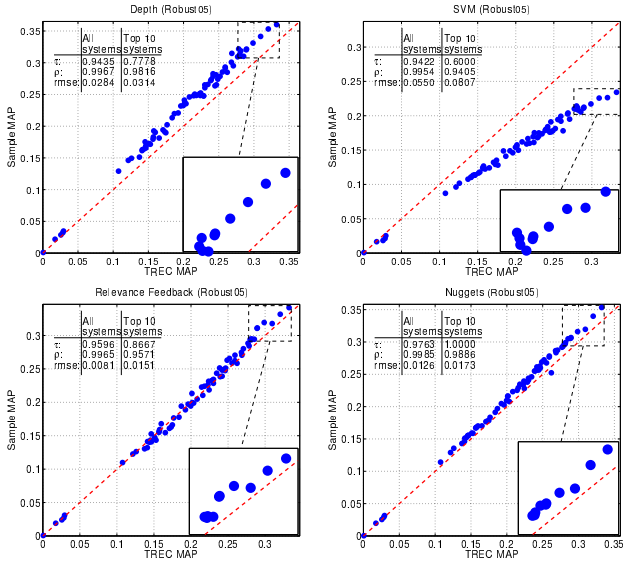

Figure 5: IR System evaluation performed by assessing 350 documents on Robust05.

representing an IR system. Perfect performance would correspond to all data points falling on the line $y = x$. Three different statistics are reported on the plot: Kendall's τ (a measurement of rank agreement), linear correlation coefficient ρ (the goodness of fit to a straight line, which implies rank correlation), and root mean squared error (the difference of estimated scores and true scores, which implies both linear and rank correlation). In most circumstances, evaluation accuracy matters most for the top systems; therefore, these statistics are separately reported both for all systems and the top 10 systems. The top 10 systems are also expanded in the plot for careful analysis. It is clear from these plots that IR system evaluation performed using the nuggets-based qrel is better that that of depth-pooling, relevance feedback and support vector machine-based qrels.

AdHoc99	Robust05	Web09
cirtrc82	indri05RdmmT	NeuRRWeb300
READWARE	rmit5t1025	THUIR09AbClu
Dm8Nbn	RIMam05l050	THUIR09QeDiv
orcl99man	juruManTit	twJ48rsU
iit99ma1	indri05RdmeD	Sab9wtBfDiv
8manexT3D1N0	indri05RdmeT	ICTNETDivR3
kdd8qe01	sab05ror1	twCSodpRNB
GE8MTD2	sab05roa2	twCSrs9N
READWARE2	uic0501	udelFMWG
kuadhoc	RIMam05l200	uwgym
932	738	672

Table 4: Systems removed for reusability experiments. Removed systems as a group bring the most unique relevant documents to the pool. AdHoc99, Robust05: pool depth is taken to be 100 documents; Web09: pool depth is taken to be 10 documents. Last row shows the total unique relevant documents removed.

Generally, the evaluations of the top 10 systems are significantly better with the nugget-based method.

3.3 Inference and Reusability

A test collection is reusable if it accurately assesses the performance of *unseen* systems, ones that did not contribute to its own construction. We can measure the reusability of a test collection by holding out a set of *seen* systems—eliminating their contribution to the test collection—and assessing the ability of the modified test collection to accurately assess both the held-out systems and the systems that yet remain.

We pick 10 systems greedily based on their high number of unique relevant documents (not retrieved by other systems). Systems removed from each dataset are listed in Table 4.

Documents Assessed	SVM	Relevance Feedback	Nuggets
AdHoc99			
100	0.3006	0.5322	0.5934
350	0.4298	0.7237	0.7448
600	0.5627	0.8057	0.8815
Web09			
100	0.4841	0.7186	0.6977
350	0.8557	0.9029	0.9036
600	0.9631	0.9184	0.9916
Robust05			
100	0.3452	0.6794	0.6614
350	0.6570	0.8462	0.8353
600	0.8576	0.8822	0.9695

Table 5: Document Relevance Inference performance (MAP), using 100, 350 and 600 assessed documents (training).

The results are shown in Figure 6. Note that we did not include the results for depth-pooling both because it is well known to suffer wildly from the exclusion of the systems with most unique relevant documents and, more importantly, it has no way to infer the relevance of the removed documents. Examining the plots, we can conclude that overall, Nugget-based test collections are more reusable than the other active learning-based approaches.

Next, we investigate the performance of the removed systems by comparing their inferred and true MAP values (the

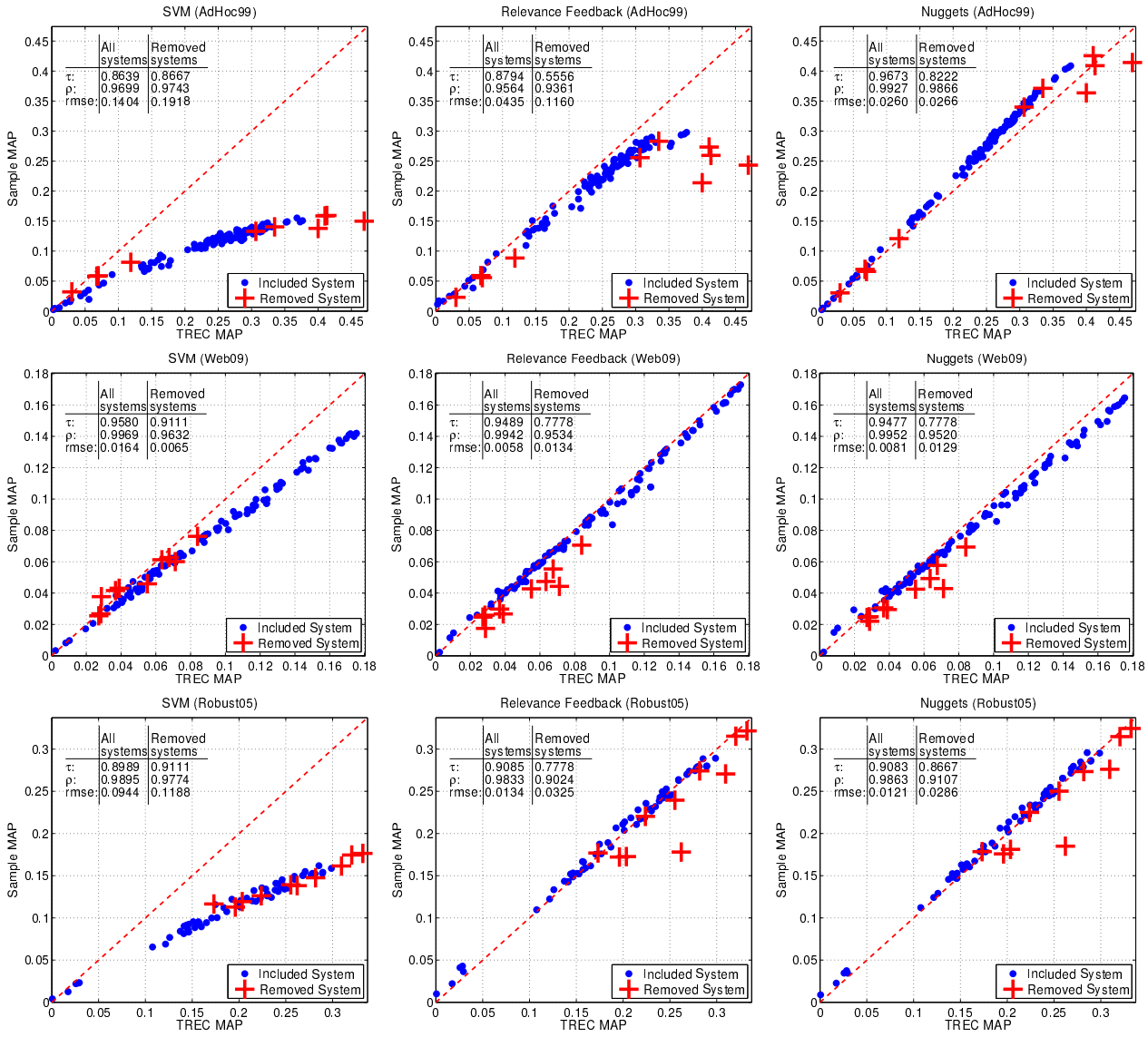

Figure 6: Resuability: IR System evaluation performed by assessing 350 documents. Top: AdHoc99; Middle: Web09; Bottom: Robust05; Left: SVM; Center: Relevance Feedback; Right: Nuggets. Red plus indicate systems removed from the dataset.

latter computed using TREC qrel). Table 6 shows the comparison when an average of about 350 documents are assessed per query. The estimated MAP values produced by the Nugget-based method are generally much close to their true MAP values than those of other methods.

We can also measure reusability by ranking all the documents by their inferred relevance scores such that the judged relevant documents are at the top of the list and those judged non-relevant are at the bottom and then computing the MAP of the list. Table 5 shows the result when 100, 350, and 600 documents are assessed per query on average; the relevance of the remaining documents are inferred and MAP value produced as described above.

4. NUGGETS QUALITY

The nuggets themselves have important practical use, but in order to rely on them, we must evaluate how well our system can separate relevant and nonrelevant nuggets. In order to measure this nugget quality, two notions are necessary at each round of the active learning process: (1) the ability to distinguish good vs bad nuggets–the Average Precision (AP) of ranked nuggets; and (2) the Recall of the nugget set "G"–the proportion of discovered nuggets which are relevant.

We conducted a large crowdsourced study to analyze these notions. First, a set of nuggets was sampled for each query, non-uniformly and without replacement, Table 7; an inclusion probability is stored for each sampled nugget such that the estimated quantities below are essentially unbiased. Each sampled nugget was then presented to 5 different assessors (Mechanical Turks), in batches consisting of 10 random nuggets and 5 random trap questions (nuggets with known relevance grades) for a single query. Binary relevance was

Removed System	True MAP	Estimated MAP RF	SVM	Nuggets
AdHoc99				
READWARE2	0.4692	0.2432	0.1499	0.4142
orcl99man	0.4130	0.2595	0.1594	0.4089
iit99ma1	0.4103	0.2734	0.1586	0.4256
READWARE	0.4000	0.2138	0.1378	0.3636
8manexT3D1N0	0.3346	0.2830	0.1404	0.3712
GE8MTD2	0.3065	0.2557	0.1329	0.3399
kdd8qe01	0.1186	0.0885	0.0814	0.1208
kuadhoc	0.0698	0.0558	0.0588	0.0662
Dm8Nbn	0.0675	0.0593	0.0582	0.0696
cirtrc82	0.0301	0.0231	0.0320	0.0303
Web09				
udelFMWG	0.0840	0.0706	0.0762	0.0694
uwgym	0.0712	0.0443	0.0601	0.0429
THUIR09AbClu	0.0676	0.0554	0.0627	0.0577
ICTNETDivR3	0.0635	0.0474	0.0613	0.0493
Sab9wtBfDiv	0.0551	0.0427	0.0459	0.0425
THUIR09QeDiv	0.0389	0.0267	0.0431	0.0295
twCSodpRNB	0.0368	0.0298	0.0416	0.0306
NeuRRWeb300	0.0287	0.0175	0.0377	0.0220
twCSrs9N	0.0284	0.0259	0.0268	0.0249
twJ48rsU	0.0272	0.0247	0.0255	0.0245
Robust05				
indri05RdmmT	0.3323	0.3216	0.1764	0.3245
indri05RdmeT	0.3204	0.3153	0.1745	0.3148
uic0501	0.3095	0.2706	0.1615	0.2761
indri05RdmeD	0.2818	0.2742	0.1475	0.2734
sab05ror1	0.2621	0.1780	0.1382	0.1850
sab05roa2	0.2553	0.2395	0.1392	0.2502
rmit5t1025	0.2238	0.2203	0.1262	0.2250
RIMam05l050	0.2035	0.1727	0.1193	0.1812
RIMam05l200	0.1961	0.1721	0.1128	0.1759
juruManTit	0.1732	0.1769	0.1165	0.1785

Table 6: Reusability: Evaluation of removed IR systems based on qrel formed by assessing 350 documents for all queries and adding inferred relevant documents to the qrel.

collected for each nugget in the batch if the assessor successfully passed the trap questions. Finally, an EM-like procedure [20, 22] was run in order to obtain reliable probabilities of relevance for each nugget by estimating the worker quality and accounting for bias[2].

For the nugget set ("G"), the recall at each round can be obtained in an unbiased fashion using the Horvitz-Thompson estimator [40]. To measure the ability of a system to separate relevant and non-relevant nuggets, we used the following ranking algorithm, independent of the nuggets sampled for assessment, and estimated Avg Precision for the obtained ranked-list of nuggets using the statAP estimator [5] on Turk-assessed nuggets. Both the Recall and the AP are estimated with high confidence [5], but we believe there is at least 5-10% nugget assessment error made by the crowdsourcing, despite using 5 assessors per nugget and a sophisticated post-assessing procedure.

The nugget ranking algorithm is a dual to that of the algorithm for ranking documents for selection and assessment. Each nugget is scored by using it to rank all documents retrieved so far by the matching score with each document and

[2]This procedure is publicly available at [21]

	Total Nuggets	Sampled Nuggets	Sampled Relev Nuggets
Adhoc99	10986.54	256	57.4
Web09	13083.48	111.48	57.38
Robust05	5872.56	93.96	26.88

Table 7: Nuggets sampled and assessed by Mechanical Turks (Query average)

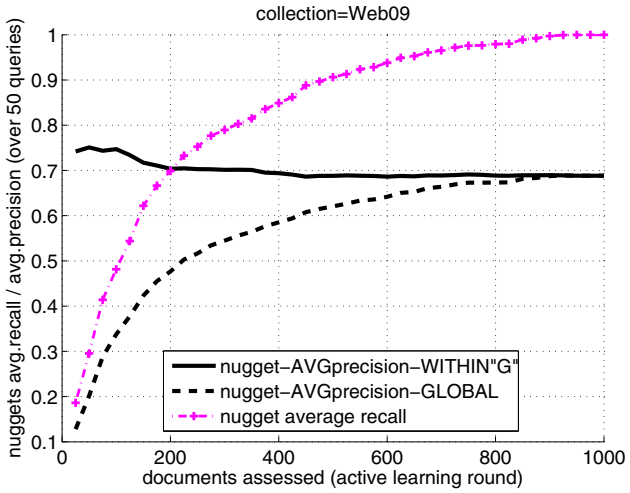

Figure 7: Nugget set Recall and Ranking Precision as a function of the documents assessed

then calculating the AP of the resulting ranked list. These scores form a ranked list of the relevance of all retrieved nuggets for a given query. We separate this into two lists: an AVGprecision-WITHIN"G", which uses the ranking of all nuggets seen so far, thus representing the ability to rank all nuggets at a given round; and an AVGprecision-GLOBAL, which includes all undiscovered and thus unranked nuggets, which represents information quality by round. These results are plotted in Figure 7 for the Web09 collection along with the average recall, the proportion of relevant nuggets which have been discovered, for that round. Note that after 300 documents are assessed, 80% of the relevant nuggets are discovered. The within-G AP is roughly constant at .7, which we believe to be an excellent value, essentially separating the relevant nuggets within the nugget set.

5. FAILURE ANALYSIS

To define the utility and limitations of our system, we analyze the conditions under which it fails to quickly achieve acceptable recall levels and the points at which documents are incorrectly selected. Due to the interconnected nature of the judgments, text extraction, and document selection, a case of an inappropriately selected document may be symptomatic of issues with one or more other documents. We therefore create a more detailed system to fully identify the sources of errors, allowing us to remedy many of the failures and to report on the limitations of the system and ideas for its further improvement.

To perform this analysis, we use heuristics to identify documents which are likely candidates for false positives and false negatives. We manually view these documents in the context of a query and system, which allows us to discern the source of the problem. We analyzed at least 5 docu-

Source of Error	Documents (%)
Assessor Disagreement	92 (33.70%)
Lack of Matching Nuggets	69 (25.27%)
Low Nugget Weights	121 (44.32%)
Extraneous Nugget Text	9 (3.30%)
Nugget Missing Keyword	7 (2.56%)
Bad Document Length Weighting	19 (6.96%)
Matching Well In Wrong Context	14 (5.12%)
Query Too General	2 (0.73%)
Incorrect Weighting	7 (2.56%)
Stopping Key Terms	2 (0.73%)
Matching Many Marginal Nuggets	27 (9.89%)

Table 8: Sources of Error for AdHoc99 DataSet.

ments per query for the AdHoc99 DataSet, totaling over 270 documents. Our findings are shown in Table 8 and described here. Note that some documents fit multiple categories, therefore the total percentage is greater than 100.

• Assessor disagreement: There are numerous cases in which a document can be considered either relevant or not, and in a number of cases we find documents with similar text in both classes.

• Lack of matching nuggets: The document did not sufficiently match nuggets, making our system unable to accurately classify it. Whether due to overly strict requirements for a match in our matching algorithm, overly general queries, or inability to match synonymous text, it was a pervasive issue in our sample. In many cases, it is also true that although the document contains relevant information, the keywords and ideas are scattered throughout the document, so no small section matches well with good nuggets.

• Low nugget weights: The weights of matching nuggets have been lowered, either due to their short length or by matching non-relevant documents, such that they do not help in selecting this document. This may occur even though the nugget appears relevant, due to the issues detailed below.

• Extraneous nugget text: By using only sentences as candidate nuggets, a nugget may contain non-relevant information in addition to the relevant portions. This non-relevant text may match documents in a different context, which are therefore non-relevant. One remedy would be to use nuggets of varying size, which is a source of our future work.

• Nugget missing keyword: The majority of a nugget may be relevant, but it is missing a keyword or phrase to separate it from non-relevant documents. This is especially an issue with pronouns.

• Bad Document Length Weighting: We attempted to use an inverse document length in the scoring function, but found it to be more harmful than beneficial in practice.

• Matching well in wrong context: As one might imagine, there are many documents which can discuss a similar subject, but may be lacking a specific entity or topic which would make it relevant to the query. However, the document may match a large portion of many good nuggets, and be incorrectly inferred as likely relevant.

• Query Too General: The matching function requires similar text between documents. With a general query, there may not be information concerning one document which matches well with information of any other document. For example, the AdHoc99 query "creativity" is a general concept with a wide variety of relevant documents. This could likely be improved with either more documents with which to match or a method for matching synonyms.

• Incorrect Weighting: Cases in which it was clear that noise in judgments caused good nuggets to be decreased in weight, or bad nuggets to increase in weight.

• Stopping Key Terms: The stopword list we use occasionally removes words which are key to a particular query, for example "Three" in the query "Three Gorges Project."

• Summing many marginal nuggets: A document can match many nuggets of small quality to a small degree, accumulating to a large nugget score.

In the cases of the small percentages, they tended to be harder issues to diagnose, but ones which may have contributed to other categories. Examples of this include the query being too general, which can cause difficulty in matching nuggets, and harmful stopping, which can prevent otherwise useful matches and lower match scores and nugget weights. Areas such as handling noise in labels and removing extraneous nugget text are planned for future study.

6. CONCLUSION

We have demonstrated a principled method of automatic nugget extraction while assessing documents, and its ability to improve the efficiency of creating test collections. Our method creates the collection qrel while minimizing the effort required by the document assessors. The nuggets based qrels were shown to produce very accurate system evaluations, but more important, shown to be reusable. This approach works on various types of documents from very different collections.

Nuggets have uses beyond IR system evaluation. They encode information relevance at a finer granularity level than documents do. As many researchers before us noted, they can be used for understanding specific behavior of users, query types, evaluate novelty/diversity/ambiguity, produce summaries, create clusters or natural subtopics, encode user cost of reading long documents.

A particular future work direction is improving the matching function by embedding NLP into our shingle-based method. We are already exploring use of look-up tables of synonyms; while better results can be obtained if we manually select the proper synonyms, we could not improve the current results using available ontologies like WordNet. Negations and coreference problems, which we believe responsible for major matching failures, requires more sophisticated NLP.

A simple way to improve matching and to filter documents is to emphasize certain keywords, related to query relevance information; we have tried this, but the only way to get such keywords is to require them from the assessor. A different assessor model is a possible direction for future research, but taking into consideration that there are quite a few things an assessor can do quickly (not time consuming): deciding different query aspects, look at nuggets with high weights and provide feedback, state keywords or synonyms etc.

Acknowledgment: This material is based upon work supported by the National Science Foundation under Grant No. IIS-1017903. Any opinions, findings and conclusions or recommendations expressed in this material are those of the

author(s) and do not necessarily reflect the views of the National Science Foundation (NSF).

7. REFERENCES

[1] *The Eighth Text REtrieval Conference (TREC-8)*. U.S. Government Printing Office, 2000.

[2] *33rd ACM SIGIR Workshop on Crowdsourcing for Search Evaluation*, Geneva, Switzerland, 2010.

[3] E. Amitay, D. Carmel, R. Lempel, and A. Soffer. Scaling IR-system evaluation using term relevance sets. SIGIR '04.

[4] J. Aslam. Improving algorithms for boosting. In N. Cesa-Bianchi and S. Goldman, editors, *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, July 2000.

[5] J. A. Aslam and V. Pavlu. A practical sampling strategy for efficient retrieval evaluation, technical report.

[6] J. A. Aslam, V. Pavlu, and R. Savell. A unified model for metasearch and the efficient evaluation of retrieval systems via the hedge algorithm. SIGIR '03.

[7] Y. Baram, R. El-Yaniv, and K. Luz. Online choice of active learning algorithms. *J. Mach. Learn. Res.*, Volume 5, December 2004.

[8] L. Bentivogli, P. Clark, I. Dagan, H. T. Dang, and D. Giampiccolo. The sixth PASCAL recognizing textual entailment challenge. In *The Text Analysis Conference (TAC 2010)*, 2010.

[9] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. 2006.

[10] A. Z. Broder. Identifying and filtering near-duplicate documents. COM '00, London, UK, 2000.

[11] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. WWW'97.

[12] B. Carterette, J. Allan, and R. K. Sitaraman. Minimal test collections for retrieval evaluation. SIGIR'06.

[13] B. Carterette and I. Soboroff. The effect of assessor error on IR system evaluation. SIGIR '10.

[14] C. L. A. Clarke, N. Craswell, and I. Soboroff. Overview of the TREC 2009 web track, 2009.

[15] C. L. A. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon. Novelty and diversity in information retrieval evaluation. SIGIR'08.

[16] G. V. Cormack, C. R. Palmer, and C. L. A. Clarke. Efficient construction of large test collections. SIGIR'98, pages 282–289, Aug. 1998.

[17] H. T. Dang, J. J. Lin, and D. Kelly. Overview of the TREC 2006 question answering track 99. In *TREC*, 2006.

[18] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. COLT '95.

[19] D. Harman. Overview of the third text REtreival conference (TREC-3). In *Overview of the Third Text REtrieval Conference (TREC-3)*. U.S. Government Printing Office, Apr. 1995.

[20] M. Hosseini, I. J. Cox, N. Milic-Frayling, G. Kazai, and V. Vinay. On aggregating labels from multiple crowd workers to infer relevance of documents. In *ECIR*, volume 7224 of *ECIR '12*, 2012.

[21] P. G. Ipeirotis. get-another-label: A set of tools for managing redundant answers collected from amazon. http://code.google.com/p/get-another-label/.

[22] P. G. Ipeirotis, F. Provost, and J. Wang. Quality management on amazon mechanical turk. HCOMP '10.

[23] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, Volume 20, 2002.

[24] T. Joachims. Training linear svms in linear time. KDD '06, pages 217–226, New York, NY, USA, 2006. ACM.

[25] S. Krenzel. Finding blurbs. Website. http://www.stevekrenzel.com/articles/blurbs.

[26] J. Lin and D. Demner-Fushman. Automatically evaluating answers to definition questions. HLT '05.

[27] J. Lin and D. Demner-Fushman. Will pyramids built of nuggets topple over? HLT'06.

[28] V. Pavlu, S. Rajput, P. B. Golbus, and J. A. Aslam. IR system evaluation using nugget-based test collections. WSDM '12.

[29] F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? CIKM '08.

[30] T. Sakai, M. P. Kato, and Y.-I. Song. Click the search button and be happy: evaluating direct and immediate information access. CIKM '11.

[31] T. Sakai, M. P. Kato, and Y.-I. Song. Overview of NTCIR-9 1CLICK. 2011.

[32] T. Sakai and C.-Y. Lin. Ranking Retrieval Systems without Relevance Assessments — Revisited. In *the 3rd International Workshop on Evaluating Information Access (EVIA) — NTCIR-8*, 2010.

[33] T. Sakai and R. Song. Evaluating diversified search results using per-intent graded relevance. SIGIR '11.

[34] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods. *The Annals of Statistics*, 26(5), 1998.

[35] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. COLT '92.

[36] I. Soboroff, C. Nicholas, and P. Cahan. Ranking retrieval systems without relevance judgments. SIGIR '01.

[37] A. Spink. Term relevance feedback and query expansion: relation to design. SIGIR '94.

[38] A. Spoerri. Using the structure of overlap between search results to rank retrieval systems without relevance judgments. *Inf. Process. Manage.*, 43, 2007.

[39] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, Mar. 1998.

[40] S. Thompson. *Sampling*. Wiley series in probability and mathematical statistics: Applied probability and statistics. 1992.

[41] E. M. Voorhees. Overview of the TREC 2005 robust retrieval track. TREC 2005.

[42] E. M. Voorhees. Question answering in TREC. CIKM '01.