Northeastern University Runs at the TREC12 Crowdsourcing Track

Maryam Bashir, Jesse Anderton, Jie Wu, Matthew Ekstrand-Abueg,
Peter B. Golbus, Virgil Pavlu, Javed A. Aslam
College of Computer and Information Science
Northeastern University, Boston, USA
{maryam,jesse,evawujie,mattea,pgolbus,vip,jaa}@ccs.neu.edu

February 6, 2013

Abstract

The goal of the TREC 2012 Crowdsourcing Track was to evaluate approaches to crowdsourcing high quality relevance judgments for images and text documents. This paper describes our submission to the Text Relevance Assessing Task. We explored three different approaches for obtaining relevance judgments. Our first two approaches are based on collecting a limited number of preference judgments from Amazon Mechanical Turk workers. These preferences are then extended to relevance judgments through the use of expectation maximization and the Elo rating system. Our third approach is based on our Nugget-based evaluation paradigm.

1 Introduction

We have participated in the Text Relevance Assessing Task (TRAT), one of the two TREC 2012 Crowdsourcing Track tasks. The goal of the TREC Crowdsourcing Track was to evaluate approaches to crowdsourcing high quality relevance judgments for text documents and images. We used the following approaches for obtaining relevance judgments:

- **pair-based ELO** The Elo rating system is a method for calculating the relative rating of players in two player games like Chess [5]. We have used this system for obtaining relevance ranking of documents instead of players by treating a preference judgment between two documents as a match between two players. The Elo rating system updates the score of a document when a certain number of preference comparisons for that document have been made. The score of every document is updated based on the score of the documents to which it was compared. In this work, we demonstrate that the Elo rating system can accurately rank documents using only $O(n)$ preference judgments.

- **pair-based EM** In this context, the Expectation Maximization algorithm [7] is a means of estimating "true" labels from crowd workers as latent variables in a model of worker quality. We convert the collected preferences into binary relevance judgments, and use EM to compute probabilities of relevance for each document.

- **nugget-based** Relevant nuggets—atomic pieces of factual information—are extracted by assessors directly from documents. The nuggets are used to find relevant documents, using a technique based on our nugget-based evaluation framework [9]. This does not use the document pair preference assessments obtained from Mechanical Turk.

1.1 Preference Judgments vs. Relevance Judgments

Traditionally, assessors are asked to give absolute relevance grades to each document with respect to some topic. However, studies have shown that assessors can give more reliable judgments if they are asked which of a pair of documents they prefer, i.e. "is document A better than document B?" rather than "how relevant is document A?" [4]. Another advantage of using preferences is that many popular learning-to-rank algorithms such as RankBoost [6] and RankNet [3] are trained on preferences. Since preferences are often not available, these algorithms need to infer preferences from the absolute relevance judgments collected from assessors. Some information is lost during this process, leading to many ties between documents. This suggests that preferences can be used to improve the training of learning-to-rank algorithms that use such a pairwise approach.

The use of preference judgments on document pairs, as opposed to absolute judgments on documents, for IR evaluations creates new challenges. There are $\binom{n}{2}$ unique pairs of documents for a list of n documents, which means that the number of judgments we need to collect increases to $O(n^2)$. Since collecting judgments is very costly, we need a mechanism for collecting these preferences efficiently. If the collected preferences were transitive, we could reduce the cost by collecting preference judgments for only $O(n \lg n)$ pairs of documents and using them as the comparator for a sorting algorithm. This would produce a sorted list of documents. Unfortunately, preference judgments are known to be not perfectly transitive. We overcome this problem by using the Elo ranking system to produce ranked lists of documents from incomplete preferences.

1.2 Relevant Nuggets vs Relevance Judgments

In a separate run, we propose a new method for relevance assessment based on relevant *information*, not relevant *documents*. Once the relevant information is collected, any document can be assessed/inferred for relevance [9]. The problem with document judgments and with its variants is that the information relevant to a topic is encoded by *documents*, and in the presence of large topic sets or large and/or dynamic collections, it is difficult or impossible to find and judge all relevant documents. Our thesis is that while the number of *documents* potentially relevant to a topic can be enormous, the amount of *information* relevant to a topic, the nuggets, is far, far smaller.

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

The rest of paper is organized as follows: Section 2 describes the collection of preference judgments on document pairs, providing details on our HIT design, experimental setup, and results. Section 3 describes the *pair-based* runs that use the Elo rating system and EM approaches. Section 4 describes our work using the Nugget-based evaluation paradigm. The final section, Section 6, concludes the paper with a description of future work.

2 Document-Pair assessment collection with Mechanical Turk

In this section we describe our experimental design for the collection of preference judgements from crowd workers and our methodology for obtaining complete relevance judgments from only $O(n)$ preference judgements. This section is organized as follows: Section 2.1 gives the details of our Amazon Mechanical Turk setup for crowdsourcing preference judgements, section 2.1.1 describes our HIT design, section 3.1 and section 5 give the methodology and results of using the Elo rating method for extracting relevance judgements from incomplete preference pairs, and section 3.2 gives the methodology and results of using EM for extracting relevance judgements from incomplete preference pairs.

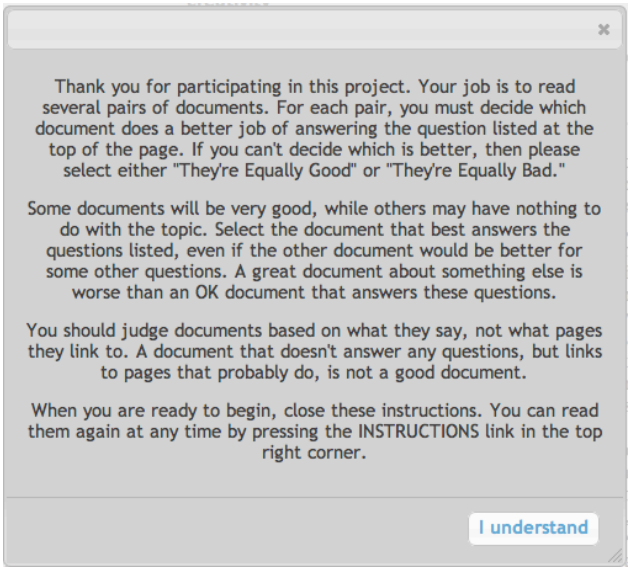

Figure 1: Instructions for preference judgements for crowd workers

2.1 Amazon Mechanical Turk HITs

Amazon Mechanical Turk (AMT) is a crowdsourcing Internet marketplace that gives developers the ability to use human intelligence for tasks that computers are currently unable to do [1]. A person or organization, termed the "requestor," creates a task definition in the marketplace which workers may then carry out in return for payment. A task definition is called a Human Intelligence Task (HIT). When a worker submits work for a given HIT, the quality of the submission can be automatically assessed and the submission can be either approved, leading to payment, or rejected. There are around 200,000 registered workers of AMT from almost 100 countries. More details on how to use this service can be found in the developer documentation [1].

2.1.1 HIT Design

The HITs we created to collect preference judgements from crowd workers had the following design. After accepting the assignment, workers were shown the instructions seen in Figure 1. In these instructions, we explained that documents should be preferred strictly based on whether they provide information about the query, description, and narrative for a particular topic: that a well-written discussion of a related topic should not be preferred to a poorly-written document which is exactly on topic.

After dismissing the instructions, the worker is shown the interface presented in Figure 2. The "Title" field of a TREC topic is displayed on top, along with its description and narrative. This information describes in detail what constitutes a relevant document for this query. Below the query information is a series of buttons, which allow workers to record their preferences. Two documents are displayed side-by-side below the buttons. The leftmost and rightmost buttons are labeled "This One," with an arrow pointing to the left or right document, respectively. These buttons allow users to choose a winning document. Between these buttons are two buttons for recording ties, labeled "They're Equally Good" and "They're Equally Bad." Each HIT consisted of 20 preference pairs for the same topic, and had a time limit of 30 minutes. Workers were paid \$0.15 for each approved HIT. The order in which the preferences were displayed, as well as the order of the documents for each particular preference, was randomized.

Document pairs were selected for judgement in the following manner. First, we calculated a prior relevance score for each document using BM25. This produced an initial ranking of the documents for each topic. For each document below the top six, we selected five documents to be compared against, uniformly at random,

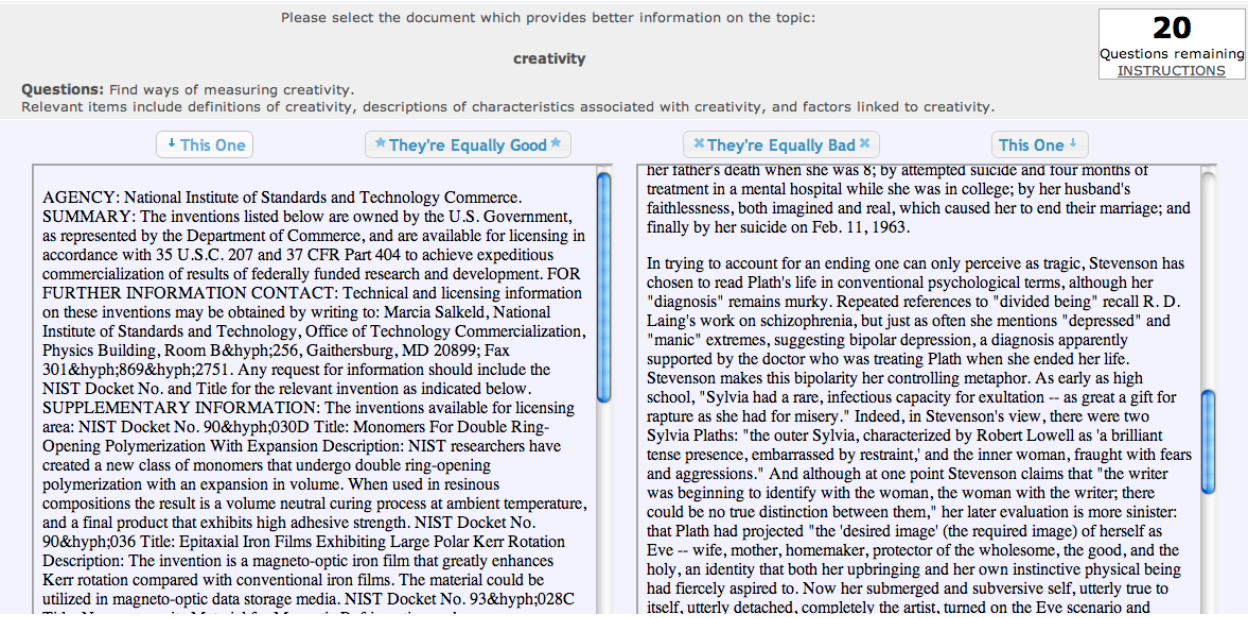

Figure 2: Preference pair selection interface with topic keywords and description

from the set of documents with higher BM25 scores. For the top six documents, we collected complete pairwise preferences. We collected four worker assessments for each preference pair which we selected for judgement.

2.1.2 Quality Control

The workers we employed have no particular training in assessing document relevance, so we need a means of verifying the quality of their work. There have been many studies on assessing worker quality for crowdsourcing platforms such as AMT [7]. We used trap questions in our study to ensure that workers are making a reasonable effort, instead of clicking randomly.

A trap question is a question inserted into the HIT which looks the same as any other, but for which a correct answer is known ahead of time. We asked five IR graduate students to create our trap questions by pairing documents which appeared to be highly relevant with documents which appeared to be highly non-relevant. We then mixed five of these trap questions, selected at random, into each HIT. As a result, each assignment consisted of five trap questions and fifteen "real" questions. A worker's submission was accepted if at least two of the five trap questions were answered correctly, and rejected otherwise.

2.1.3 Data

There were a total of 18,260 documents from TREC 8 ad-hoc, which used the Text Research Collection Volumes 4 (May 1996) and 5 (April 1997) minus the Congressional Record (CR). 10 topics were selected randomly from TREC topics. Participants of the crowdsourcing track were required to simulate the role of NIST assessors for the 10 TREC 8 ad-hoc topics.

3 Pair-based runs

3.1 The Elo Rating System

The Elo rating system is a method for calculating the relative rating of players in two player games [5]. The rating system assigns each player a rating score, with a higher number indicating a better player. Each player's rating is updated after he or she has played a certain number of matches, increasing or decreasing in value depending on whether the player won or lost each match, and on the ratings of both players competing in each match—beating a highly rated player increases one's rating more than beating a player with a low rating, while losing to a player with a low rating decreases one's score more than losing to a player with a high rating. These scores are used in two ways: 1) players are ranked by their scores, 2) the scores can be used to compete the likelihood of each player beating every other player. If the matches are selected intelligently, this can be accomplished even if only $O(n)$ matches are played.

For our problem, we treat each document as a player and each preference judgment between two documents as a match between players. All documents enter the "tournament" rated equally. After each document "plays" a certain number of matches, we update each document's rating according to equation 3. After all the matches are played, we can rank the documents by their final score. This list can be thresholded to produce absolute relevance judgments. We can also use the scores to compute transitive preference judgments.

For our preliminary experiments, we select $O(n)$ matches stochastically. We wish to sample pairs in such a way that we create a bias towards relevant documents. In this way, relevant documents will play more matches than non-relevant documents, giving them more opportunities to improving their ratings and move up the list. We begin by ranking documents for each topic using the BM25 retrieval method. The first five documents all "play" one another. Each remaining document in the list plays against 5 randomly selected documents with higher ranks. In our experiments, every document plays at least 5 matches. On average, every document plays 11 matches. We sort all documents based on their ratings after all $O(n)$ matches have been played.

3.1.1 Mathematical Details of The Elo Rating System

The Elo rating system measures the skill level of players in relative terms. Each player's rating depends on the rating of his or her opponents, as well as the of wins or losses. The expected score for each player in a match can be estimated from the ratings of the players. If player A has rating R_A and player B has rating R_B, then the expected score of player A is calculated as follows:

$$E_A = \frac{1}{1 + 10^{\frac{R_B - R_A}{200}}} \tag{1}$$

Similarly the expected score for player B is:

$$E_B = \frac{1}{1 + 10^{\frac{R_A - R_B}{200}}} \tag{2}$$

Player's ratings are updated by comparing their actual score to their expected score. When a player performs worse than expected, then his or her rating is decreased. If player A has an expected score, E_A, and an actual score, S_A, then his or her rating will be updated by following formula:

$$R'_A = R_A + K(S_A - E_A) \tag{3}$$

This update can be made after every match, or after several matches according to the situation.

ELO Experimental Setup. We have set K equal to 20 in Elo rating update equation 3 and each document has an initial uniform rating of 1000. Every document pair selected for comparison was judged by 5 crowd

workers. For each of the 10 topics, participants of TREC Crowdsourcing track were required to provide both binary relevance judgments and probabilities of relevance for each document. Probabilities were produced by normalizing the Elo scores. We then label the n most probable documents as relevant, where n was chosen by manual inspection.

3.1.2 TREC Runs "NEUElo2/3/4/5"

We have submitted four different runs to the TREC Crowdsourcing track using the Elo rating algorithm for preference judgments. Due to time constraints we were not able to collect five judgments for every document pair for our submitted runs to TREC. The following is a description of each of the runs created using the Elo rating algorithm:

NEUElo2: This run was created using a non-uniform number of judgments for each document pair, i.e. the number of judgments for each document pair could be from 1 to 5. The threshold for binary relevance was the top 20 documents for each topic.

NEUElo3: This run was the same as NEUElo3 except that we also used the judgment from our trap questions. Since the trap questions were created by experts, we added the documents preferred in the trap questions to the top of each ranked list. We also chose a different number of relevant documents for topic, again by manual inspection.

NEUElo4: This run was the same as NEUElo3 except that we ignored all the document pairs that had only 1 judgment. This was done to avoid random judgments.

NEUElo5: This run was the same as NEUElo4 except that the threshold for binary relevance was the top 20 documents for each topic.

3.2 Expectation Maximization run "NEUEM1"

Expectation Maximization (EM) is an iterative method for finding the maximum likelihood estimate of the parameters of a probability distribution using a model incorporating unobserved latent variables. In our model, we treat the observed preference judgments from Crowdsource workers as being drawn from a distribution parameterized by the "quality" of each worker. The relevance grade of each document is treated as a latent variable. For the purposes of this task, the end result is a probability distribution over each latent variable, i.e. the probability that each document is relevant. Our approach is similar to that of Hosseini et al. [8].

3.2.1 The EM Algorithm

Assume that there are N documents and M crowd workers. Let G represent the number of relevance grades. We assign each worker j a $G \times G$ latent confusion matrix, C^j. Each element c^j_{lg} is the probability that worker j assigns a label l to a document whose true relevance is g. In our model, each document has one of two relevance grades, relevant and non-relevant; i.e. $g \in \{r, n\}$. Hence C^j is a 2×2 confusion matrix, $C^j = \begin{Vmatrix} c_{rr} & c_{rn} \\ c_{nr} & c_{nn} \end{Vmatrix}$. Let $\Pr(R_i = g)$ denote the probability that document i has the true relevance grade g, p_g is the prior probability distribution over relevance grades, and X^j_{il} is an indicator variable indicating whether worker j has assigned labeled l to document i.
There are five steps in the EM algorithm:

Step 0: Pre-processing

We wish to learn the relevance grades of each document. However, we have collected preferences between pairs of documents. Before we can use the EM algorithm, we must first convert these preferences into grades. We do so in the following way: assume that worker j prefers document A to document B. Then we interpret this as worker j labeling document A as relevant, and document B as non-relevant.

Step 1: Initialization

We assume that all of the crowd workers are honestly attempting to give correct answers. Therefore, we initialize each workers confusion matrix to:

$$C_j = \begin{Vmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{Vmatrix}$$

Furthermore, we assume that all documents are equally likely to be relevant and non-relevant. Therefore, we initialize $p_r = 0.5$ and $p_n = 0.5$.

Step 2: Estimate the probability of relevance

In this step, we combine the confusion matrix and relevance labels from each worker to estimate the probability that each document is relevant. All labels are assumed to be i.i.d.

$$\Pr(R_i = r \mid C_j, \forall j \in M) = \frac{p_r \times \prod_{j=1}^{M} \prod_{l=0}^{G} {c_{lr}^j}^{X_{il}^j}}{p_r \times \left(\prod_{j=1}^{M} \prod_{l=0}^{G} {c_{lr}^j}^{X_{il}^j} + \prod_{j=1}^{M} \prod_{l=0}^{G} {c_{ln}^j}^{X_{il}^j} \right)} \tag{4}$$

$$\Pr(R_i = n \mid C_j, \forall j \in M) = \frac{p_r \times \prod_{j=1}^{M} \prod_{l=0}^{G} {c_{ln}^j}^{X_{il}^j}}{p_r \times \left(\prod_{j=1}^{M} \prod_{l=0}^{G} {c_{lr}^j}^{X_{il}^j} + \prod_{j=1}^{M} \prod_{l=0}^{G} {c_{ln}^j}^{X_{il}^j} \right)} \tag{5}$$

Step 3: Estimate the maximum likelihood

Using the each documents' estimated probability of relevance calculated in step 2, we produce an updated estimate of each worker's confusion matrix.

$$c_{lg}^j = \frac{\sum_{i=1}^{N} \left(\Pr(R_i = g) \times X_{il}^j \right)}{\sum_{k=0}^{G} \sum_{i=i}^{N} \left(\Pr(R_i = g) \times X_{ik}^j \right)} \tag{6}$$

Step 4: Iterate Step 2 and 3 until convergance

We repeatedly calculated worker confusion matrices and probabilities of document relevance until the results converged. We consider this process to have converged when, for each document i, the difference between $\Pr(R_i = g)$ at iteration $t-1$ and at iteration t for all documents i and relevance grades g is less than or equal to 0.01.

After applying the steps above, we created ranked lists by sorted the documents by their probability of relevance, i.e. $\Pr(R_i = r)$. Additionally, all documents whose probability of relevance is greater than 0.5 is marked as relevant.

4 Nuggets-based run "NEUNugget12"

Separate from pair assessments runs using mechanical turk, we produced a run using our nugget technology [9]. We use a framework of mutual, iterative reinforcement between nuggets and documents, most similar to Active Learning [10, 2] mechanisms. The human feedback (relevance assessment) is given *iteratively* on the documents/data; thus, while the assessor judges the document (as it is standard IR procedure), our "back-end" system infers the relevance of unjudged documents as well as the relevance of text fragments, or nuggets, from within the documents.

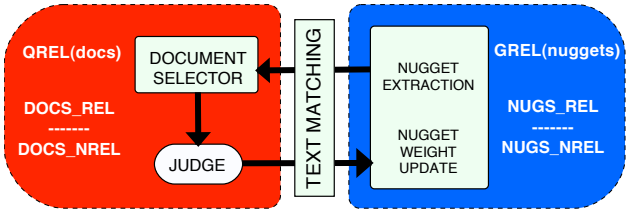

Figure 3: The overall design of assessment process: Iteratively, documents are selected and assessed, and nuggets are extracted and [re]weighted.

Figure 3 illustrates this framework, with 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 GREL); documents are judged and put into the set of documents QREL;
- for relevant documents, nuggets are extracted and added to GREL
- all nugget scores are updated based on the relevant and non-relevant documents judged so far and how well each nugget matches them

This framework uses four components that can be designed somewhat independently. Our implementa-

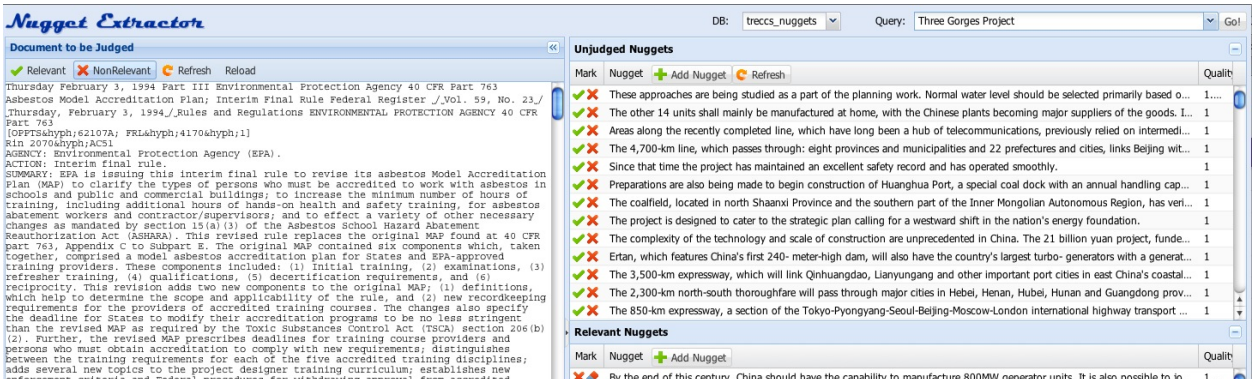

Figure 4: The nugget extraction interface, on query "Three Gorges Project", showing a document to be judged (left) and the list of candidate nuggets (right)

tions for SELECTION, MATCHING, and EXTRACTION are presented in [9], 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.

A fifth 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.

The supervised aspect of the nugget extraction process is shown in Figure 4. Here the user views the next document to be judged as well as the current relevance beliefs of all extracted nuggets. One aspect of the process added to our previous work is the ability to mark nuggets as relevant and nonrelevant. This is achieved through the buttons next to each nugget on the right side of the interface. While not necessary, if a user judges a nugget, we can integrate this information into our update system.

Every nugget has a score that is a combination of the relevance judgments of all matching judged documents. A matching relevant document increases the score, and a nonrelevant document decreases the score. Therefore, if a nugget is marked relevant, we set its score to the maximum of all unjudged nuggets, and if a nugget is marked nonrelevant, it is given the minimum score of all unjudged nuggets. As described in our previous paper, these scores are then used to update the relevance belief of unjudged documents, and to then select the next documents for judgment. Finally, when the process stops (usually up to the assessor), to obtain binary relevance judgments from document scores, we use a threshold of 0.8.

This resulted in a low amount of manual work per query, as a small percentage of documents were judged and the relevance of the rest were extrapolated. The entire nugget extraction process took about one hour per query and was performed by a grad student. While only one person was used, the methodology could utilize judgments from multiple workers to reduce user error. We performed our experiments with one person to demonstrate the small amount of manual work required to infer relevance grades for the entire set.

5 Results

Following the conventions of the TREC Crowdsourcing track, we evaluate binary preference judgements using both Logistic Average Misclassification (LAM) and the area under the ROC curve (AUC). The evaluation results of our runs submitted to TREC are given in Table 1.

Topic ID	NEUElo2		NEUElo3		NEUElo4		NEUElo5		NEUEM1		NEUNuggets12	
	AUC	LAM	AUC	LAM	AUC	LAM	AUC	LAM	AUC	LAM	AUC	LAM
411	0.665	0.293	0.69	0.11	0.693	0.145	0.692	0.171	0.650	0.383	0.860	0.121
416	0.735	0.265	0.745	0.209	0.854	0.183	0.855	0.162	0.623	0.432	0.868	0.220
417	0.678	0.261	0.695	0.195	0.716	0.182	0.716	0.156	0.728	0.196	0.686	0.132
420	0.579	0.285	0.598	0.134	0.642	0.153	0.639	0.151	0.524	0.464	0.865	0.157
427	0.747	0.146	0.756	0.134	0.788	0.122	0.788	0.12	0.605	0.260	0.798	0.191
432	0.598	0.38	0.598	0.256	0.631	0.38	0.631	0.38	0.624	0.405	0.580	0.375
438	0.578	0.272	0.583	0.38	0.62	0.319	0.617	0.311	0.578	0.154	0.762	0.447
445	0.775	0.232	0.783	0.12	0.706	0.144	0.706	0.212	0.691	0.452	0.891	0.121
446	0.683	0.392	0.72	0.227	0.729	0.238	0.711	0.156	0.656	0.460	0.707	0.274
447	0.78	0.265	0.953	0.012	0.992	0.051	0.953	0.082	0.652	0.186	0.464	0.081
All	0.682	0.279	0.712	0.178	0.737	0.192	0.731	0.19	0.633	0.339	0.748	0.213

Table 1: Evaluation results of runs submitted to TREC 2012 Crowd Sourcing track.

6 Conclusion and Future Work

In this paper we have described our work based on three approaches for obtaining high quality relevance judgements. The first two approaches are based on crowd sourcing relevance judgements using preferences,

whereas the third approach is based on nuggets assessments.

Elo runs use partial preference judgements obtained from crowd workers as input to a popular rating system that outputs a rank list of documents with their relevance scores. Our preliminary results submitted to TREC crowd sourcing track are close to average results submitted to TREC by various systems for most of the topics. We later have experimented with some modifications to out basic Elo rating system, and collected more crowdsourced pair assessments, which resulted in significant improvement to our reported TREC 2012 results. For our future work, we plan to design more intelligent matches between documents, with a goal to obtain high quality judgements from small number of preference judgements.

The EM run is a straight forward application of Expectation Maximization algorithm on crowdsourced assessments, as other researchers have done, in order to account for worker quality. We use it as a baseline comparison, but we think it can be most useful as an intermediary processing stage of crowd workers assessments, before running other methods like Elo. One of the Elo enhancements mentioned above is to run it over EM-processed pair assessments.

The nuggets run is very different than the others, to the point that "crowdsourcing" might not be an appropriate name: the focus here is not on cheap weak judgments that can be averaged somehow into a good assessment, but rather on very few judgments of high quality (on nuggets) and some expert feedback on documents. We note that the nuggets run did best among the runs we submitted; while not a lot better, the human effort put in to it, even at expert level, is significantly less than either crowdsourcing or TREC assessing efforts. Unfortunately, later analysis showed that our "expert" grad student performing the assessments disagreed significantly, perhaps mistakenly, with the existing TREC qrel assessments, thus certainly impacting the run performance.

References

[1] Amazon mechanical turk. `http://www.mturk.com`.

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

[3] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, ICML '05, pages 89–96, New York, NY, USA, 2005. ACM.

[4] B. Carterette, P. N. Bennett, D. M. Chickering, and S. T. Dumais. Here or there. In *ECIR*, pages 16–27, 2008.

[5] A. Elo and S. Sloan. *The Rating of Chess Players, Past and Present.* Arco Publishing, New York, NY, USA, 1978.

[6] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4:933–969, Dec. 2003.

[7] M. Hosseini, I. J. Cox, N. Milić-Frayling, G. Kazai, and V. Vinay. On aggregating labels from multiple crowd workers to infer relevance of documents. In *Proceedings of the 34th European conference on Advances in Information Retrieval*, ECIR'12, pages 182–194, Berlin, Heidelberg, 2012. Springer-Verlag.

[8] G. Kazai, J. Kamps, M. Koolen, and N. Milic-Frayling. Crowdsourcing for book search evaluation: impact of hit design on comparative system ranking. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, pages 205–214, New York, NY, USA, 2011. ACM.

[9] S. Rajput, M. Ekstrand-Abueg, V. Pavlu, and J. A. Aslam. Constructing test collections by inferring document relevance via extracted relevant information. In *Proceedings of the 21st ACM Conference on Information and Knowledge Management*, pages 145–154. ACM Press, October 2012.

[10] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the Fifth Annual AC¡ Conference on Computational Learning Theory*, COLT '92, 2002.