Optimizing Nugget Annotations with Active Learning

Gaurav Baruah[1], Haotian Zhang[1], Rakesh Guttikonda[1],
Jimmy Lin[1], Mark D. Smucker[2], Olga Vechtomova[2]

[1] David R. Cheriton School of Computer Science
[2] Department of Management Sciences
University of Waterloo

{gbaruah, haotian.zhang, rguttiko, jimmylin, mark.smucker, ovechtomova}@uwaterloo.ca

ABSTRACT

Nugget-based evaluations, such as those deployed in the TREC Temporal Summarization and Question Answering tracks, require human assessors to determine whether a nugget is present in a given piece of text. This process, known as nugget annotation, is labor-intensive. In this paper, we present two active learning techniques that prioritize the sequence in which candidate nugget/sentence pairs are presented to an assessor, based on the likelihood that the sentence contains a nugget. Our approach builds on the recognition that nugget annotation is similar to high-recall retrieval, and we adapt proven existing solutions. Simulation experiments with four existing TREC test collections show that our techniques yield far more matches for a given level of effort than baselines that are typically deployed in existing nugget-based evaluations.

1. INTRODUCTION

In the context of the Cranfield Paradigm [11], we have witnessed the evolution toward more fine-grained batch evaluation methodologies for assessing system output. Examples include passage retrieval [1], aspect retrieval [20], and the nugget evaluation methodology [19, 3]. This paper focuses on the nugget annotation phase of the nugget evaluation methodology, where assessors determine if a piece of text contains a particular nugget.

Nuggets represent atomic facts relevant to users' information needs, usually expressed as short natural language phrases. Given a set of nuggets, which can be viewed as an "answer key", assessors must read all system output and determine which (if any) nuggets are present—this is called nugget annotation. Once the nuggets have been identified, it is straightforward to compute a final metric based on the matches. It bears emphasizing that the nuggets represent concepts, and thus the nugget matching process requires understanding the *semantics* of the system output—taking into account synonyms, paraphrasing, and other linguistic phenomena. For example, the nugget "the train crashed at the buffer stop" might manifest in system responses as "slammed into the end of the line", "smash into a barrier", and "hit the barrier at the end of the platform". A baseline evaluation workflow, and indeed one that is typically deployed in existing implementations of nugget evaluation methodologies, requires assessors to exhaustively consider all system outputs with respect to all nuggets.

The insight behind our work is the recognition that the process of nugget annotation can be viewed as a high-recall retrieval task, where system outputs represent the "document collection" and the nuggets represent the "topics". Thus, we can exploit recently-developed active learning techniques that have been validated for high-recall retrieval to the nugget annotation problem. The additional wrinkle, however, is that each topic contains multiple nuggets that all (eventually) need to be found. Thus, the question is—in what order shall we "search" for the nuggets? Our answer is to let the active learning technique (the learner) itself tell us where to look next.

The main contribution of this paper is the novel adaptation of a state-of-the-art active learning technique for high-recall retrieval to the problem of nugget annotations. We developed two different matching strategies for letting the learner guide the assessor as to which system output to examine next and what nuggets are likely to be found there: in the greedy approach, the learner simply proposes the system output most likely to contain a nugget (i.e., a nugget-sentence pair is presented as a candidate match for annotation); in an alternative approach, we attempt to maximize the likely number of candidate matches presented to the user. We apply our proposed techniques to simulate the nugget annotation process using four large-scale evaluations from TREC, the QA tracks from 2004–2005 and the Temporal Summarization tracks from 2013–2014. Experimental results show that our techniques yield far more matches for a given level of effort than baselines that are typically deployed in existing nugget-based evaluations.

2. BACKGROUND AND RELATED WORK

2.1 Nugget-Based Evaluations

We begin with a more precise formulation of the problem. In nugget-based evaluations, for a particular information need (i.e., a topic), we have an "answer key" comprising $|N|$ nuggets (expressed as short natural language phrases). We assume that system output is discretized into some unit of text (in our case, sentences), and there are $|D|$ such units across all the system outputs we are trying to evaluate. The

CIKM'16 , October 24 - 28, 2016, Indianapolis, IN, USA

ISBN 978-1-4503-4073-1/16/10…$15.00

DOI: http://dx.doi.org/10.1145/2983323.2983694

test collection	# topics	#identified nuggets |N|	#pooled sentences |D|	match matrix |N| x |D|	avg. #ngts / topic	avg. #upds / topic	#confirmed matches	avg. #conf. matches/topic	uniq. #ngts in matches
TS13	9	1,077	9,113	1,283,298	119.7	1,012.6	4,850	538.9	484
TS14	15	1,394	14,651	1,407,448	92.9	976.7	13,635	909.0	837
QA04	64	580	61,642	471,881	9.1	963.2	3,840	60.0	545
QA05	75	758	496,535	4,981,989	10.1	6,620.5	4,159	55.5	687

Table 1: Statistics of the four TREC test collections used in this study. For confirmed matches, exact duplicates were removed from the judged set of sentences.

nugget annotation process is to complete an $|N| \times |D|$ matrix with boolean values: true if the nugget is contained in the text and false if not. As discussed above, this matching requires considering the meaning of the system output. Evaluation results are then straightforwardly computed as a function of this match matrix, although for the purposes of this work, the actual metrics are not important, and therefore it is not necessary to keep track of which $d \in D$ was produced by which system. We make the additional simplifying assumption that the cost of nugget matching is a constant, even though the unit of system output (sentences) might vary in length and judging some nuggets may require more cognitive load than others.

Note that for exhaustive evaluation, we must fill the entire match matrix. It would, however, make sense to prioritize the examination of match candidates based on the likelihood that it contains a nugget. This is especially valuable in a scenario where resources are constrained, as it would represent a more efficient use of assessor effort. We do exactly this, guided by active learning techniques.

Four large-scale implementations of the nugget evaluation methodology include the TREC Question Answering (QA) tracks in 2004–2005 [18, 19] and the TREC Temporal Summarization (TS) tracks in 2013–2014 [2, 3]. For the TS tracks, systems were required to return sentences that are relevant and timely with respect to specified newsworthy events (topics). The QA tracks required systems to return responses (sentences) to questions regarding an event or entity. For both, the relevance of returned sentences was measured in terms of nuggets contained in them. We used data from all these evaluations to validate our work. To quantify the amount of effort involved in the nugget annotation process, Table 1 lists the number of nuggets identified by assessors for the various tracks. For example, in TS13, there was an average of 119.7 nuggets and 1012.6 sentences per topic with $|N| \times |D| = 1,283,298$ total matches across all topics. Assessors took 15 hours on average per topic for the matching process for TS13 [10].

For the TS tracks, the annotation process was aided by an assistive user interface [2] that lists all sentences to be judged and all nuggets, in two columns. For each relevant sentence, an assessor annotated parts of the sentence that match the corresponding nugget. A nugget can match more than one sentence and a sentence can contain more than one nugget. For QA evaluations, assessors took a similar approach.

We note that researchers have developed automatic techniques for matching nuggets against system output based on n-gram overlap, e.g., POURPRE [13] and Nuggeteer [14] in the context of QA. These techniques are useful for rapid formative evaluations, but we believe that human review is still necessary for summative evaluations such as TREC.

To be clear, our work tackles only the nugget annotation process and assumes the existence of a nugget "answer key". These nuggets, however, must come from somewhere: in TREC QA, the nuggets were identified during topic development as well as derived manually from the pooled system output by assessors [18], which was a labor-intensive process. Pavlu et al. [15] manually extracted nuggets from returned documents and used features of identified nuggets to prioritize assessment of additional documents. Rajput et al. [16] utilized an active learning feedback loop where "nuggetization" and assessments are interleaved by automatically weighting sentences within documents as candidate nuggets. In this work we tackle the more established workflow where the creation of the nuggets is distinct from the nugget annotation process itself; e.g., in TREC TS, the nuggets were first extracted from Wikipedia articles corresponding to news events that comprised the topics [3].

2.2 Technology-Assisted Review

Technology-assisted review (TAR) applies iterative retrieval to help reviewers identify as many relevant documents as possible from a collection in the context of electronic discovery (eDiscovery). Cormack and Grossman [9] developed an autonomous TAR (AutoTAR) protocol by extending continuous active learning (CAL) for TAR. In CAL, the most likely relevant documents identified using a machine learning method are interactively presented to reviewers for assessment. Application of CAL for TAR maximizes recall while requiring less review effort than comparable methods [8].

AutoTAR was implemented as the baseline model implementation (BMI) for the TREC 2015 Total Recall Track [17]. It initially ranks the entire document collection through a classifier trained on an initial query (treated as a pseudo-document) and 100 randomly selected documents, which are assumed to be non-relevant. The user is asked to assess the most relevant (i.e., highest scoring) document, and then retrains the classifier on this new assessment along with the query and 100 new random (assumed non-relevant) documents. In an ideal situation, AutoTAR would repeat this process, selecting the top document for assessment and then retraining on all available assessments. Since this is not computationally practical, AutoTAR requests assessments in exponentially increasing batches, starting with a batch size of one and increasing the batch size by $\textsc{Max}(1, 0.1 \cdot$ `batch_size`$)$ each iteration.

The TREC 2015 Total Recall overview paper [17] indicates that the BMI was hard to beat, i.e., no automatic or manual run performed consistently better than the BMI. Given the similarities between identifying relevant documents in a collection and identifying matching nuggets in system responses (both of which require high recall), we adapted AutoTAR to our problem, described next.

3. NUGGET ANNOTATION STRATEGIES

For matching sentences with nuggets, we envision an assistive user interface that presents the assessor with *match*

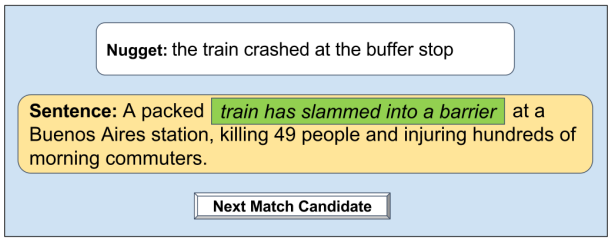

Figure 1: Simulated assessment interface for nugget annotation. An assessor reads the sentence and the nugget, annotates the substring that represents the nugget (if there is a match), and moves on to the next match candidate.

candidates—a nugget and a sentence that is likely to contain the nugget (see Figure 1). The assessor judges whether the sentence indeed contains the nugget and appropriately *annotates* the match (i.e., records a representation of the nugget in the sentence), or rejects the match. The assessor then moves on to the next match candidate. This is repeated until all matches are found or some stopping criterion (e.g., the assessor runs out of time or the evaluation runs out of resources). We present a number of nugget annotation strategies that select the next match candidate for presentation.

3.1 Baselines

Let us consider the match matrix (Section 2) as having rows that represent nuggets and columns that represent sentences. One obvious nugget annotation strategy is to consider each sentence in turn and look to see if each nugget is found in it; this strategy can be described as a *column-by-column* (CBC) matching strategy. A converse strategy is to present a nugget at a time to the assessor and match all sentences against the nugget. Here, the item under scrutiny is the nugget and the matches are made *row-by-row* (RBR). The choice of sentence or nugget to be assessed next is arbitrary. Thus, RBR and CBC represent baselines, and indeed quite close to the approaches that have been deployed by previous large-scale evaluations.

3.2 Proposed Techniques

Although the total number of valid matches is the same across strategies, depending on the matching algorithm, as well as the assessment interface, the effort required for complete assessment may vary. Given the various combinations possible, we describe a nugget annotation strategy as the combination of an assistive user interface along with a specific active learning algorithm for matching. We develop two matching strategies and compare them with the CBC and RBR baselines. Each matching strategy takes as input the set D sentences pooled for a topic and the set N nuggets that were identified for the topic. Both strategies follow the general outline of the AutoTAR BMI deployed for the TREC Total Recall track, described in Section 2.2, with one key difference: we train one classifier for each nugget instead of one classifier for the entire topic.

Most Likely Candidate (MLC). The Most Likely Candidate matching strategy presents the match candidate (sentence s_i, nugget n_k) to the assessor that is most likely to result in a positive match, i.e., the likelihood of nugget n_k being present in sentence s_i is the highest across *all* all possible match candidate pairs. Initially, a separate classifier is constructed for each nugget in a given topic using the relevant features. Then, all sentences in D are classified according to each nugget's classifier and the scores are recorded, i.e., a classification score is produced for each position in the match matrix. The (s_i, n_k) pair with the highest global classification score is presented to the assessor for review. The classifier for n_k is retrained after the assessor either confirms (annotates the nugget representation in the sentence) or rejects the match. Once assessed, the match candidate pair (s_i, n_k) is not presented again for assessment. When five match candidates have been assessed, all sentences are rescored against the newly-trained classifiers for each nugget. Since we have multiple classifiers for each topic, we do not retrain the classifiers in increasing batch sizes as in the BMI; instead, we retrain immediately but we recompute sentence scores periodically after a set number of assessments. The process is repeated until a sufficient number of valid matches are identified.

Cover Maximum Candidates (CMC). The Cover Maximum Candidates (CMC) matching strategy tries to prioritize for assessment sentences that are likely to have multiple matching nuggets. That is, CMC selects sentences that potentially match multiple nuggets. We extend the assessment interface as envisioned in Figure 1 to display M nuggets that are likely to match the sentence.

As in MLC, one classifier is trained for each nugget in a topic. Then, classification scores are computed for the entire match matrix. The sentence s_i that likely matches the most number of nuggets (with classification scores greater than a threshold ϵ) is chosen for assessment. First, all candidate matches that score above a threshold ϵ are shortlisted from the match matrix. Then, from the shortlisted candidate matches, the sentence s_i that is likely to match the maximum number of nuggets is selected. Finally, sentence s_i and the top M likely matching nuggets are presented to the assessor for review. The assessor either confirms or rejects each of the M match candidates and the respective nuggets' classifiers are accordingly retrained. All sentences are rescored against the newly-trained classifiers for each nugget after M match candidates are assessed. The process is repeated until a sufficient number of valid matches are found. If there are no scores above ϵ in the match matrix, then we fall back to MLC.

Although the choice of M for CMC is arbitrary, presenting a relatively small number of nuggets at a time serves two purposes: (i) the assessor deals with minimal information overload, and (ii) the match candidates can be displayed on the screen without the need to scroll or paginate. In our simulation experiment, one sentence along with at most $M = 5$ nuggets are putatively displayed in the assessment interface. As the probability of presenting non-matching candidates increases in later stages of the assessment process, assessment effort may increase accordingly. We therefore set a global threshold ϵ for filtering the likely match candidates. Without a threshold ϵ or a presentation limit M, we risk falling back to CBC from CMC, since TS topics routinely contain upwards of 50 nuggets. Presenting a large number of nuggets (large M), may cause errors as assessment fatigue sets in over time. For our experiments we use logistic regression classifiers for AutoTAR (Section 5) and through manual observation of the match candidates' classification scores, we set a threshold of $\epsilon = 1$ (corresponding to a match proba-

bility of 0.73). We found that scores above this setting were more likely to be valid matches across all test collections.

Another advantage of using a threshold and presentation limit is that secondary assessors (who are not topic originators) may benefit by having limited information presented to them at any given time. It is possible to tune ϵ and M dynamically according to the status of the classifiers at various stages of assessment, and set optimal values via cross-validation for each test collection, an exercise we leave for future work.

4. EVALUATION METHODOLOGY

Evaluation effort can be measured in a variety of ways, for example, in terms of time or mental energy spent to perform a particular evaluation task (e.g., to judge a document). In nugget annotations, not only do assessors have to judge if a sentence is relevant (i.e., contains a nugget), but they must also provide support by annotating the text content corresponding to the nugget. It is therefore necessary to quantify the effort for various components of the assessment process explicitly. The amount and types of user interactions with the nugget annotation interface differ across matching strategies, but in general, the total effort of using an assessment interface to perform an evaluation can be estimated as the sum of the effort of each interaction [12, 4]. For simplicity, we model effort in abstract units instead of real-world quantities such as time. Of course, the actual effort may be estimated by observing assessor behavior or by utilizing available information about the reading speed of users [6].

We designed our assessment interface (Section 3) to incur negligible memory load on the assessor by explicitly presenting all information needed to reach a decision. In the case of CBC, RBR, and MLC, the assessor considers only a nugget and a sentence at a time. In the case of CMC, the interface eliminates scrolling by presenting a small, fixed number of nuggets to the assessor.

Our basic effort model has the following components:

1. The effort of reading the sentence, λ_{read};
2. The effort of rendering a judgment about the match, λ_{match} (which includes annotating the nugget representation in the sentence if necessary).

Given the effort associated with these actions, we can compute the cost of making a single match as:

$$\text{effort} = \lambda_{read} + \lambda_{match}. \quad (1)$$

The total cost of the CBC, RBR, and MLC strategies is the sum of the effort required for each match candidate. For simplicity, we assume unit cost, i.e., $\lambda_{read} = \lambda_{match} = 1$.

For CMC, we need a slightly different effort model. Here as well, reading the sentence incurs effort λ_{read}. Then, considering the first matching nugget incurs effort λ_{match}, as before. However, considering additional nuggets may not incur the full matching effort. This is because after considering the first match, the assessor already has the semantic content of the sentence in short-term working memory, and so considering additional nuggets might not take as much effort. Let us call the effort associated with each subsequent match $\lambda_{match'}$. We currently lack empirical evidence as to what the proper setting of this effort should be, but for these experiments we consider the values $\lambda_{match'} = \{0, 0.25, 0.5, 0.75, 1\}$. In the first case, subsequent matches are "free" (i.e., zero effort), and in the last case, each additional match is just as expensive as the first. These seem like reasonable bounds for effort in practice. Thus, for CMC, we arrive at the following (where $M = 5$):

$$\text{effort} = \lambda_{read} + \lambda_{match} + (M-1) \cdot \lambda_{match'} \quad (2)$$

Note that in this model for CMC, we assume that each subsequent nugget match has constant cost.

5. EXPERIMENTS AND RESULTS

Our experiments were conducted using the nugget-based test collections from the TREC QA04, QA05, TS13, and TS14 tracks. Each track provides, for each topic (i) a set of nuggets, (ii) a pool of sentences from system output, and (iii) a mapping of nuggets to the assessed sentences (i.e., qrels). It is therefore possible to use these data to evaluate our nugget annotation strategies via simulation, based on the qrels. Table 1 presents statistics for each dataset. Note that since matches were not found by NIST for all nuggets, we restricted our experiments to only those that have at least one confirmed match in the system outputs (Table 1; column 10). Also, we observed many duplicate sentences in the judged set for each track. Although adding duplicates of judged sentences for evaluation does not significantly affect the relative effectiveness of systems [5], for our experiments, we removed all exact duplicates and keep only one copy to reduce assessment effort. Finally, we note that QA05 is substantially larger than the other collections in terms of the number of pooled sentences.

Following the AutoTAR process, our implementations use tf-idf Porter-stemmed word features and the Sofia-ML logistic regression classifier.[1] We retained digits as features since numbers are heavily used in nuggets. We removed terms with a document frequency of one due to their sparsity in order to prune the feature space. For our simulation, a given sentence is matched with a particular nugget only once.

5.1 Detailed Topic Analysis

We begin with a detailed analysis of a single topic (topic 10 from TS13) to give the reader an intuition of the behavior of our techniques. This particular topic has 521 unique (non-duplicate) sentences and 27 nuggets, with a total of 349 matches.

Figure 2 shows the matches vs. effort plot comparing MLC to the CBC and RBR baselines. For each curve, each point represent a nugget match and color is used to identify individual nuggets (i.e., all points with the same color represent the same nugget). The slope of the curves can be interpreted as the rate at which we're finding matches—i.e., the steeper the slope, the more nuggets we're finding given a unit of effort. The RBR matching strategy considers each nugget in turn, and cycles through all sentences (with an arbitrary ordering of the nuggets). The CBC matching strategy considers each sentence in turn, and cycles through all nuggets. We see that the MLC strategy switches between nuggets since it proposes the most promising candidate each time—for example, the long run of nugget matches in brown for RBR near the end is prioritized much earlier because it represents a sequence of "easier" and more likely matches.

[1] https://code.google.com/archive/p/sofia-ml/

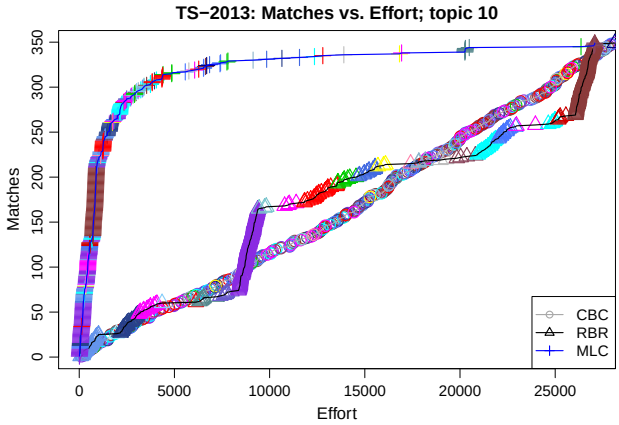

Figure 2: Illustration of how MLC switches between different nuggets, based on the most likely match.

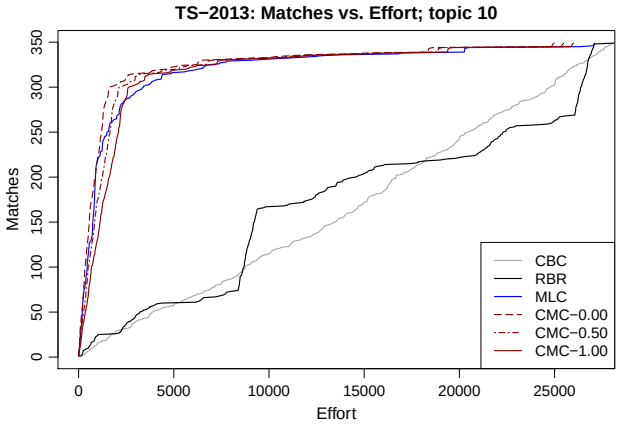

Figure 3: Comparsion of CMC, MLC, and the baselines (CBC and RBR) for TS13 topic 10.

Overall, we clearly see that for a given level of effort (x-axis), the MLC strategy finds more matches.

Figure 3 shows the performance of the CMC strategy in comparison with the MLC, CBC and RBR strategies on the same topic (we now remove the point markers for clarity). Recall that CMC presents multiple nuggets that are likely to be contained in the same sentence for assessment (Section 3.2). Matching the first nugget to the sentence incurs full effort. However, as we discussed, consideration of subsequent nuggets may not require full effort, since the assessor already holds the sentence in working memory (Section 4). Unsurprisingly, if we model the effort of matching additional nuggets as zero, $\lambda_{match'} = 0$ in the curve CMC–0.0, then we see that it is the most efficient of all strategies. On the other hand, if matching additional nuggets incurs the full cost of the first match, $\lambda_{match'} = 1$ in the curve CMC–1.0, then the strategy is less efficient than MLC, at least in the initial stages of the simulation. For the intermediate values, $\lambda_{match'} = \{0.25, 0.75\}$, the curves lie between those of $\lambda_{match'} = \{0.0, 1.0\}$, and therefore are not shown for brevity. In this particular topic, the relative performance of MLC vs. CMC depends on the level of effort and the setting of $\lambda_{match'}$, but the overall differences are relatively minor.

5.2 Aggregate Performance

To gauge the performance of a matching strategy across multiple topics in a test collection, we first normalize, for each topic, the effort required for finding all matches using RBR to the range [0,1]. The amount of effort required by the other strategies are then scaled relative to the total RBR effort; note that total CBC effort is the same as the RBR effort. The number of matches is similarly scaled to the range [0,1], which can be interpreted as recall. We then perform a 101-point interpolation to plot the match-recall vs. effort for each strategy. The results are shown in Figure 4 for each test collection: TREC QA from 2004 and 2005, and TREC TS from 2013 and 2014. These curves show the effort expended for a particular strategy averaged across all topics, relative to the maximum effort expended when using the RBR strategy. Note that all figures have the same scales, with the exception of QA05, which has substantially more pooled sentences than the rest (see Table 1).

We find that MLC performs slightly better than CMC on average when $\lambda_{match'} = 1$, i.e., the CMC(–1.0) curve, at least in the early stages of the simulation. We might consider this an upper bound on effort, as matching additional nuggets is likely to be cheaper in practice. On the other hand, $\lambda_{match'} = 0$, i.e., CMC(–0.0) should be considered a lower bound on effort, as it makes an unrealistic assumption on assessment effort. We suspect that the real answer lies somewhere between these two curves, but we note that there is no appreciable difference between the different strategies across all the test collections. However, it is clear that MLC and CMC are much more efficient in finding nugget matches than the RBR and CBC baselines.

6. FUTURE WORK AND CONCLUSIONS

The starting point of this paper is the observation that nugget annotations can be viewed as a high-recall retrieval problem, where system outputs are the "documents" and the nuggets represent the "topics". We have successfully adapted the AutoTAR, a highly-effective baseline from the TREC Total Recall tracks, to tackle this problem. We show that by training a separate classifier for each nugget in a topic, we can let active learning techniques guide the assessment process. Although both our proposed variants are virtually indistinguishable in terms of performance, simulation experiments show that our approach is much more efficient than existing baselines.

There are, however, a number of limitations to our study. We currently use a crude effort model measured in abstract units: actual effort in practice should be measured in terms of physical units such as time, and depends on many additional factors we have not accounted for such as assessor expertise. A more accurate model must also take into account assessor fatigue and learning effects, as our effort parameters are unlikely to remain constant throughout a real evaluation. Calibration against observations of actual human behavior is necessary to further validate our model.

Another issue we have yet to address is the question of when to stop. One simple answer would be to stop when a pre-allocated amount of effort has been expended, but it might be desirable to stop when the assessment process reaches the point of diminishing returns. For this, we might also look at solutions developed for high-recall retrieval [7].

Nevertheless, despite these shortcomings, we note that refinements and improvements on our techniques are un-

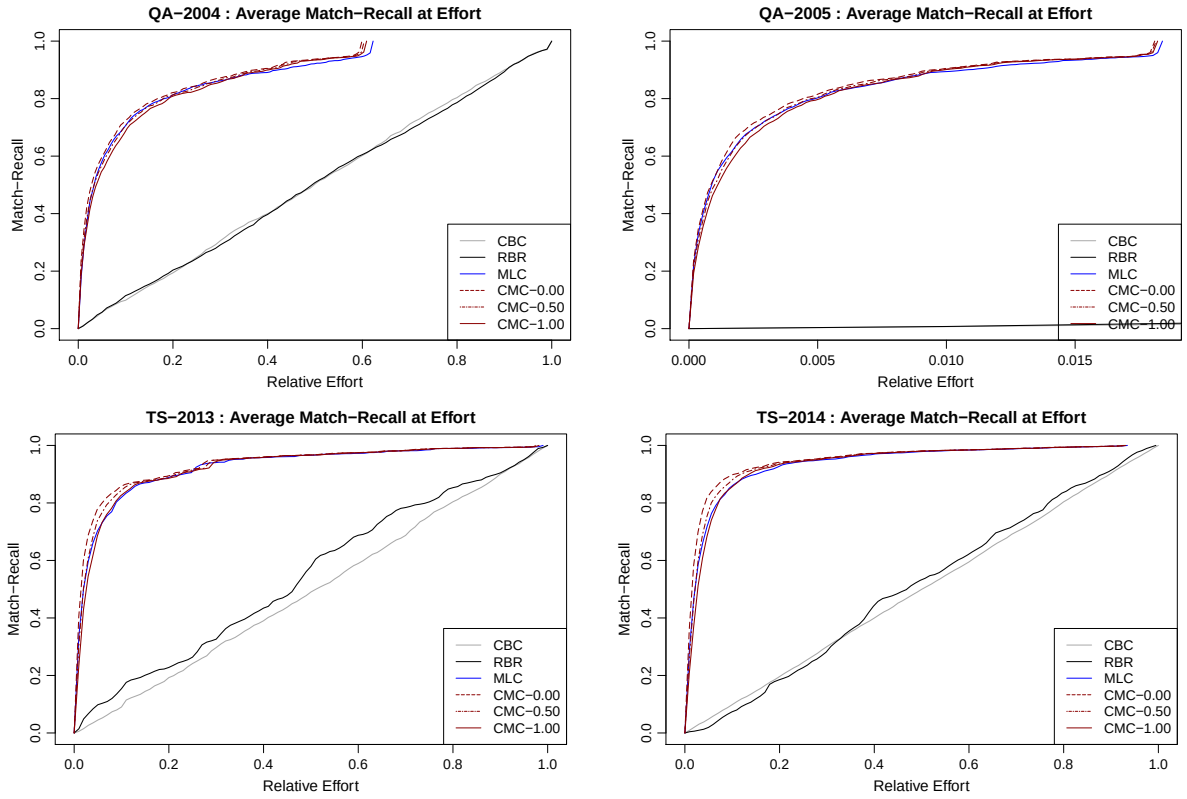

Figure 4: Average interpolated results: QA04 (top, left), QA05 (top, right), TS13 (bottom, left), and TS14 (bottom, right). Note that all graphs are on the same scale except for QA05.

likely to alter the finding that active learning techniques can reduce the effort associated with nugget annotations. Future nugget-based evaluations should consider deploying the types of techniques that we propose here.

7. ACKNOWLEDGMENTS

We would like to thank Leif Azzopardi for helpful discussions and Adam Roegiest for comments on earlier drafts of this paper. This work was made possible by the facilities of SHARCNET (www.sharcnet.ca) and Compute Canada, and was supported in part by NSERC, in part by a Google Founders Grant, and in part by the University of Waterloo.

8. REFERENCES

[1] J. Allan. HARD Track Overview in TREC 2004 High Accuracy Retrieval from Documents. *TREC*, 2004.

[2] J. A. Aslam, M. Ekstrand-Abueg, V. Pavlu, F. Diaz, and T. Sakai. TREC 2013 Temporal Summarization. *TREC*, 2013.

[3] J. A. Aslam, M. Ekstrand-Abueg, V. Pavlu, F. Diaz, and T. Sakai. TREC 2014 Temporal Summarization. *TREC*, 2014.

[4] L. Azzopardi and G. Zuccon. Building and Using Models of Information Seeking Search and Retrieval: Full Day Tutorial. *SIGIR*, 2015.

[5] G. Baruah, A. Roegiest, and M. D. Smucker. The Effect of Expanding Relevance Judgements with Duplicates. *SIGIR*, 2014.

[6] C. L. A. Clarke and M. D. Smucker. Time Well Spent. *IIiX*, 2014.

[7] G. Cormack and M. Grossman. Engineering Quality and Reliability in Technology Assisted Review. *SIGIR*, 2016.

[8] G. V. Cormack and M. R. Grossman. Evaluation of Machine Learning Protocols for Technology Assisted Review In Electronic Discovery. *SIGIR*, 2014.

[9] G. V. Cormack and M. R. Grossman. Autonomy and Reliability of Continuous Active Learning for Technology Assisted Review. *CoRR*, abs/1504.06868, 2015.

[10] M. Ekstrand-Abueg. Personal Communication. 2014

[11] D. Harman. Information Retrieval Evaluation. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 3(2), 2011.

[12] J. He, M. Bron, A. de Vries, L. Azzopardi, and M. de Rijke. Untangling Result List Refinement and Ranking Quality: A Framework for Evaluation and Prediction. *SIGIR*, 2015.

[13] J. Lin and D. Demner-Fushman. Automatically Evaluating Answers to Definition Questions. *NAACL-HLT*, 2005.

[14] G. Marton and A. Radul. Nuggeteer Automatic Nugget Based Evaluation Using Descriptions and Judgements. *NAACL-HLT*, 2006.

[15] V. Pavlu, S. Rajput, P. B. Golbus, and J. A. Aslam. IR System Evaluation using Nugget-based Test Collections. *WSDM*, 2012.

[16] S. Rajput, M. Ekstrand-Abueg, V. Pavlu, and J. A. Aslam. Constructing Test Collections by Inferring Document Relevance via Extracted Relevant Information. *CIKM*, 2012.

[17] A. Roegiest, G. Cormack, M. Grossman, and C. Clarke. TREC 2015 Total Recall Track Overview. *TREC*, 2015.

[18] E. M. Voorhees. Overview of the TREC 2004 Question Answering Track. *TREC*, 2004.

[19] E. M. Voorhees. Overview of the TREC 2005 Question Answering Track. *TREC*, 2005.

[20] E. M. Voorhees and D. K. Harman. *TREC: Experiment and Evaluation in Information Retrieval*, MIT press Cambridge, 2005.