UWaterlooMDS at the TREC 2017 Common Core Track

HAOTIAN ZHANG[1], MUSTAFA ABUALSAUD[1], NIMESH GHELANI[1], ANGSHUMAN GHOSH[1], MARK D. SMUCKER[2], GORDON V. CORMACK[1], and MAURA R. GROSSMAN[1],

[1] David R. Cheriton School of Computer Science, University of Waterloo

[2] Department of Management Sciences, University of Waterloo

In this report, we discuss the experiments we conducted for the TREC 2017 Common Core Track. We implemented a High-Recall retrieval system for assessors to efficiently find relevant documents within a limited period of time. The system consists of an AutoTAR Continuous Active Learning (CAL) system based on paragraph/document level relevance feedback. The system also includes a search engine where users can repeatedly enter their own queries to find relevant documents. For the first round of submissions, we used our system to judge 32,172 documents across 250 topics. The system's trained model of relevance was used to rank all unjudged documents. After each judgment, our system can retrain the machine learning model and rank the whole collection while maintaining interactive performance without any user discernible lag. For the second round of submissions, we crafted a user study to measure the impact of interaction mechanisms on technology assisted review and completed an initial version of the study with 10 users and 50 topics.

1 INTRODUCTION

The task of TREC 2017 Common Core Track [1] is an ad-hoc retrieval of documents for a set of topics from the New York Times dataset [2]. Manually reviewing documents and labeling relevance always requires a lot of human effort. The objective of our system is to assist assessors in finding as many relevant documents as possible within limited time. In other words, the target is to achieve high recall with reasonable effort.

High-Recall retrieval addresses the problem of finding as many relevant documents as possible with reasonable effort in terms of time or budget. One example task is building a test collection for IR evaluation. Interactive Searching and Judging (referred to as "ISJ") is one of the effective methods to build test collections [4]. In ISJ, assessors repeatedly reformulate new queries and review top ranked results from search engines. Compared to traditional pooling method, ISJ can generate comparable gold standard with less effort [4, 10, 15].

AutoTAR — an implementation of Continuous Active Learning ("CAL") protocol [4, 5] applies relevance feedback and machine learning to find the most-likely relevant documents for assessment. The TREC Total Recall Track 2015 and 2016 evaluated different retrieval methods with a simulated human-in-the-loop process to achieve high recall [7, 9]. A baseline model implementation ("BMI") based on AutoTAR was provided and served as the baseline model in Total Recall Track 2015 and 2016. None of the submitted runs were able to consistently outperform BMI [7, 9, 19]. The BMI also showed its effectiveness on other tasks such as systematic review for evidence based medicine [8] and nugget annotation [1].

Recently, Zhang et al. [17] used sentence-level feedback instead of document-level feedback in CAL to achieve high recall. Their results showed that by using sentence-level feedback only, achieved the same recall as using document-level feedback, at the same amount of effort (number of judgments). If effort was measured by the number of sentences reviewed, Zhang et al. [17] suggest using sentence-level feedback was far more efficient than document-level feedback.

However, sentence-level feedback on CAL as suggested by Zhang et al. [17] was only evaluated inside a simulation study and not by human assessors. The system was running over a predefined complete label set [9] and all aspects of human interactions were simulated. For the TREC Common Core Track, we designed a system

[1] https://github.com/trec-core/2017

[2] https://catalog.ldc.upenn.edu/ldc2008t19

for the human assessors to efficiently find and assess documents. According to the two deadlines made by TREC Common Core track, we designed a two step experiment. We first implemented a prototype system and judged documents for 250 topics by ourselves. After verifying the effectiveness and efficiency of the prototype system, we improved the prototype system according to our own user experience and implemented an improved system. Then 10 users were recruited to use the improved system to judge 50 NIST topics.

We designed our system with the following objectives:

- **Effectiveness**: The assessors should be able to easily find relevant documents using our system. Our system combines the advantages of ISJ and CAL.
- **Efficiency**: Our system applies the relevance feedback from assessor to retrain the machine learning model. Therefore it requires that the machine learning model to be retrained and the entire collection to be re-ranked after each judgment while maintaining interactive performance with no user discernible lag. Assessors should be able to quickly judge the document without the need to review its full content. The paragraph-length document summary and keyword highlighting features were provided to assist assessment.

Our group (UWaterlooMDS) implemented an assessment system that included a search engine and a CAL-based component. Using our system, we manually judged documents for all the 250 topics. In addition to judging documents ourselves, we also crafted a user study and recruited 10 users to use different variants of our improved assessment system to judge 50 NIST topics. In total, we submitted 10 manual runs to TREC.

ALGORITHM 1: The AutoTAR algorithm

Step 1. Find a relevant "seed" document using ad-hoc search, or construct a synthetic relevant document from the topic description;
Step 2. The initial training set consists of the seed document identified in step 1, labeled "relevant";
Step 3. Set the initial batch size B to 1;
Step 4. Temporarily augment the training set by adding 100 random documents from the collection, temporarily labeled "not relevant";
Step 5. Train an Logistic Regression classifier using the training set;
Step 6. Remove the random documents added in step 4;
Step 7. Select the highest-scoring B documents for review;
Step 8. Review the documents, coding each as "relevant" or "not relevant";
Step 9. Add the documents to the training set;
Step 10. Increase B by $\lceil \frac{B}{10} \rceil$;
Step 11. Repeat steps 4 through 10 until a sufficient number of relevant documents have been reviewed.

2 ASSESSMENT SYSTEM

We implemented a document assessment system with an online front-end interface accessible through a web browser. The system was composed of two components: **Search model** and **CAL model**. Each model has its own interface that was used to assess documents. The search model was built to allow users to query and search for documents to judge. While in the CAL model, users were automatically presented with documents to assess.

The search model consists of a search engine based on Indri [13] with the same retrieval and snippet generation algorithms mentioned in Smucker et al. [12]. In the search interface, users can enter their own queries and the top ranked documents retrieved by the search engine were shown to the users. Each document included a title, a snippet, and three buttons for judging. The three buttons corresponded to three relevance categories: *Highly-Rel, Relevant, Non-Relevant.* Users can click on any of these buttons to make a judgment or change an

existing judgment. Users can also click on a document to view its full content, search its content and make a judgment.

The CAL model is a variation of BMI. It returns a batch of most likely relevant documents to the user for further review. Each time a user made a new judgment, the relevance judgment was used to retrain the machine learning model, and a new batch of documents was returned and the next most-likely relevant document was shown to the user for assessment. Judgments from both search interface and CAL interface were used to retrain the CAL model. In the CAL interface, users can undo and change their previous judgments. Modified judgments are also used to retrain the CAL model.

Users were not limited to a single interface and can switch to any of the two interfaces at any moment. In both interfaces, users were able to search and highlight desired keywords in the a document. Any word (or part of a word) entered in the highlighting search bar is highlighted if it exists in the document's body or title. In the CAL interface, the highlighting persists across successive documents. To make the process of judging documents faster and more efficient, we introduced 3 keyboard shortcuts that corresponded to the three relevance categories. Users can choose to click on any of the judging buttons to make a judgment.

Two versions of the system were implemented for the two different submission deadlines of the TREC Common Core track. For the first submission, a prototype system described in the next section was built. After the first deadline, we made several improvements based on our experience with using the prototype system. The improved system was then applied in a controlled user study. Judgments collected from the user study were used to compose runs for the second submission deadline.

2.1 Prototype system for the first submission

Our UWaterlooMDS team used the prototype system to find and judge documents for all the 250 topics. The judgments collected from the prototype system were used to compose three runs that were submitted to the TREC Common Core track. Each component of the system is described in more details in the following subsections.

2.1.1 Search Model. The search interface is shown in Figure 1. Users can enter their own queries in the search bar. In the prototype system, search engine only retrieved the top 20 results for a given query. Pagination in the search interface is not supported. Each result in the SERP (Search Engine Results Pages) is composed of a document title, id and a snippet. To help users refine their search result, we allowed the following search operators:

- Double quotes (**“”**): to search for an exact phrase or word. For example, *“New York Times”*.
- Plus sign (+): to require the presence of a word in the search result. For example, *+France*.
- Combination of + and **“”**: to require the presence of phrases in the search result. For example, *+“New York Times”*.

Three judgment buttons: “Relevant”, “On topic” and “Not relevant” are provided next to each result. In the prototype system, the relevance categories are:

- **Relevant**: If the document directly addresses the core information need of the topic and covers all the aspects of the topic.
- **On topic**: If the document only mentions partial information of the topic and does not address the information need in the topic's description or its narrative.
- **Not relevant**: If the document is not relevant to the topic.

Users can click on any of the judgment buttons from the SERP to make a judgment. Clicking on any of the search result displayed a popup containing the full document. The full document popup is shown in Figure 2. As shown in the figure, the title and the content of the document are present, along with the same judgments buttons for judging the document. The popup automatically closes once a judgment is made. Users can use the “Ctrl +

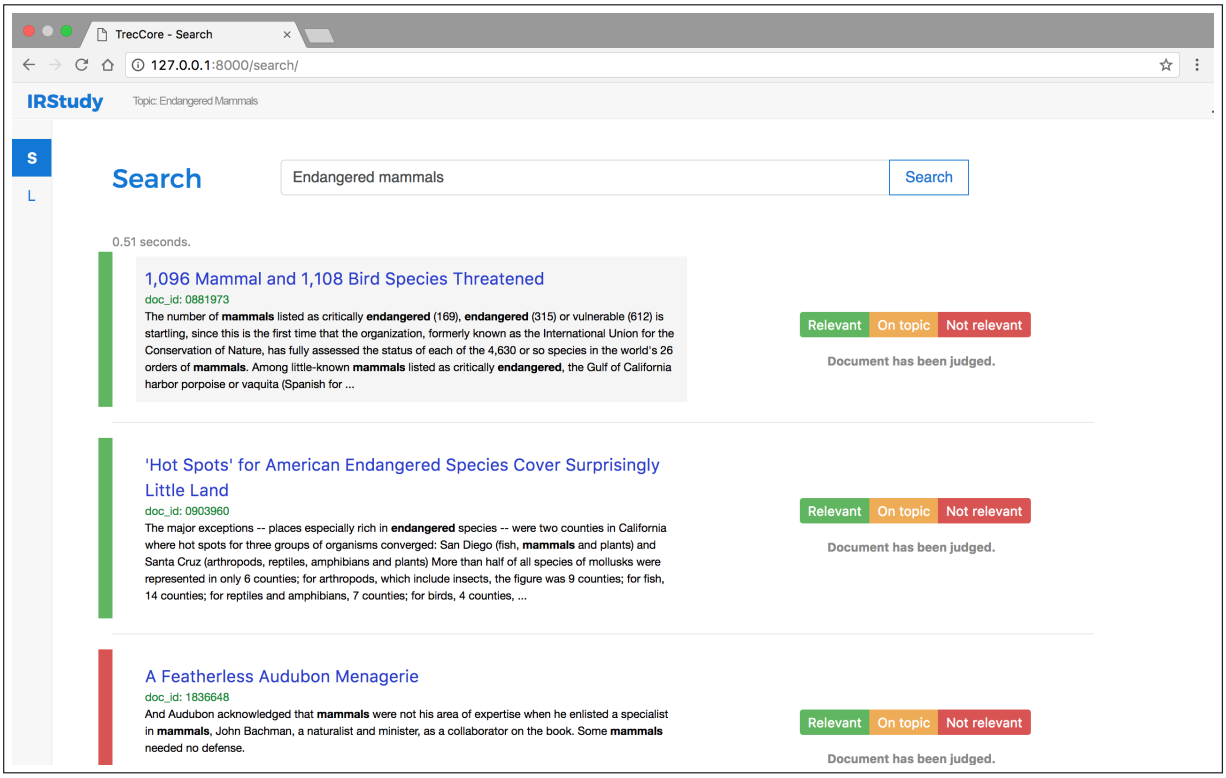

Fig. 1. Search Interface layout in prototype system. The first two results shown in the SERP are "Relevant", and third result is "Not relevant".

F" shortcut or enter any keywords in search box to highlight keywords in the document. Multiple keywords separated by space can be entered to highlight each keyword simultaneously.

The SERP may contain documents which have already been judged. These documents are visually labeled with their current judgments as shown in Figure 1. Users may rejudge these documents.

2.1.2 CAL model. Unlike the search model where users have to enter their own queries and search for documents, the CAL model automatically selected the most-likely to be relevant document and presented it to the users for assessments. The CAL model is a Continuous Active Learning system, which is based on the AutoTAR baseline model implementation ("BMI") [3]. The AutoTAR algorithm is described in Algorithm 1. The original BMI algorithm was written in *BASH*, which is suitable for simulations but inefficient for practical use. We rewrote the BMI algorithm in C++ and made several improvements and changes for our prototype system:

- **Seed query**: In BMI, the syntactic seed query for the initial classifier is the title of the topic. In our user study, the seed query was entered by the users themselves. Users could enter any terms they believed to be helpful and related to the topic.
- **Batch size**: In BMI, the batch size increases exponentially. We propose a real-time re-ranking CAL system in our setting. The system re-trains the model and re-ranks all the documents every time a new judgment is available from the user.

[3]https://plg.uwaterloo.ca/ gvcormac/trecvm/

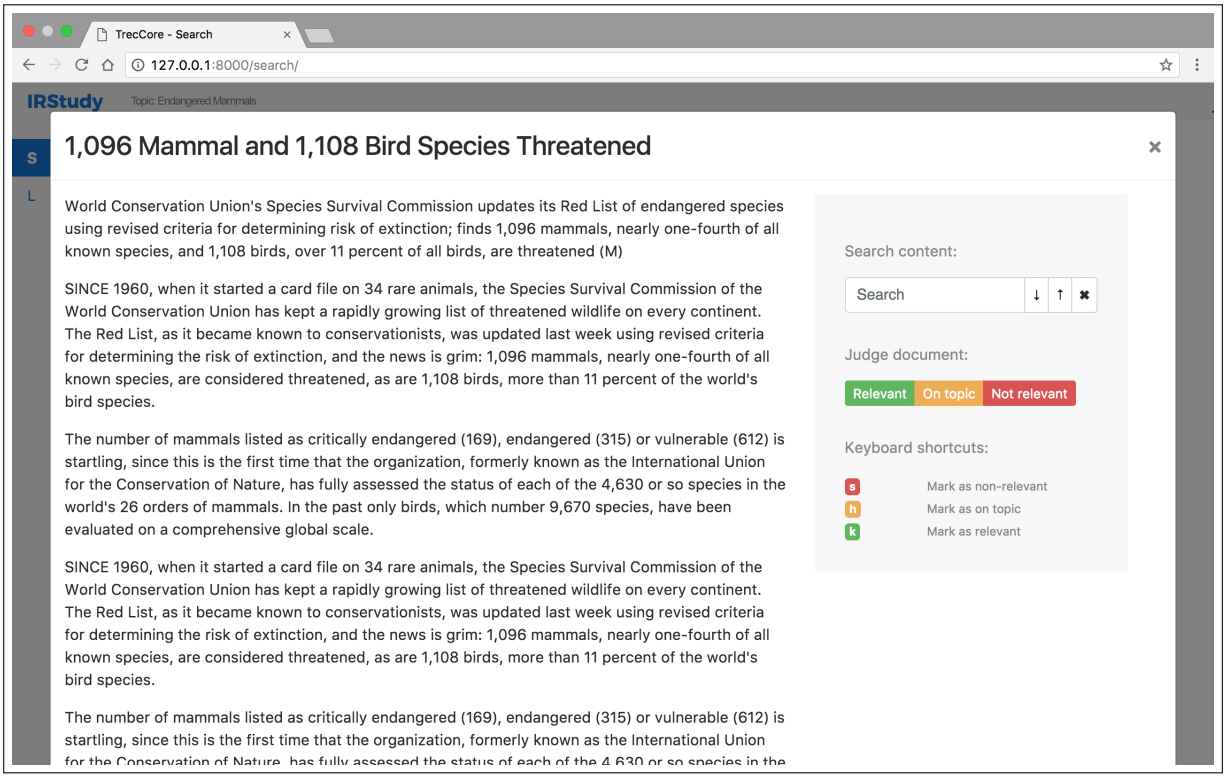

Fig. 2. Clicking at any result from Search Interface will show the full document judgment interface.

- **In memory processing**: Re-ranking 1.8 million documents is computationally expensive, especially when it is done after every judgment. A significant part of that computation is the score calculation which is the inner product of a document feature vector and the model feature vector. To enable fast re-ranking, we store all the document feature vectors in memory, and parallelize the computation across documents.
- **Documents Cache Queue**: The process of re-ranking may cause noticeable delay for users. Instead of waiting for the ranking, we immediately show the user the next highest ranked document from the current ranking. We maintain a queue of 10 documents in the order of their rank. This queue is flushed and updated with a new ranking as soon as it is received from the back-end server.
- **Logistic Regression**: was done using Sofia-ML[4] with the same hyperparameters as the BMI.

The judgment interface for the CAL model is shown in Figure 3. The interface is similar to the full document popup view in the search interface but has the following differences:

(1) We provided a document summary above the full document content. Snippet generation was based on terms weighted by the CAL logistic regression model. Terms that are more relevant and informative to a particular task get higher weights. We identified the top 10 weighted terms.

Snippets are generated using the paragraph ranking algorithm described in section 9.6 in [2]. The algorithm selects a portion or a cover $[u, v]$ of a document such that it contains at least k query terms. Multiple such covers may exist and we find all of them. We also collect covers that contain at least two of

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

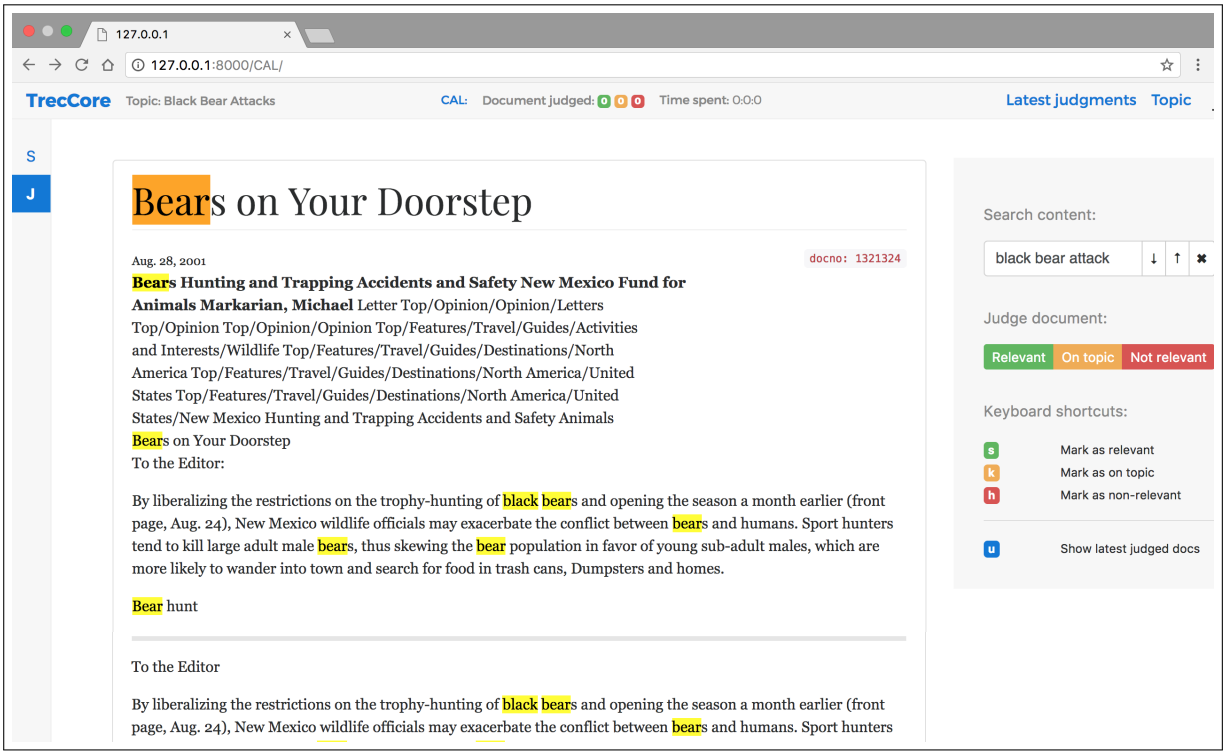

Fig. 3. CAL interface in the prototype system.

the query terms, by iterating over covers that contain 2 up to n terms where n is the number of terms in the given query.

A snippet s of length l produced by query q is scored as follows:

$$p(t) = \frac{l_t}{l_c} \tag{1}$$

$$p(t, l) = 1 - (1 - p(t)^l) \tag{2}$$

$$score(s, q) = \prod_{t \in q} p(t, length(s)) \tag{3}$$

Here, $p(t)$ gives the prior probability for the appearance of a term t at any position in the corpus (l_t is the frequency of term t, whereas l_c is the total number of words in the corpus). Thus, $p(t, l)$ gives the probability of a snippet of length l containing all the terms in the query q. A sufficiently large snippet would contain all words with a high probability. Thus, a shortest snippet s containing all the terms were ranked the lowest, and represents the case where it was least likely to have all the terms, but still does. This makes the snippet interesting to us.

To create a summary, we selected the paragraph that contained the lowest scoring snippets. In case snippet generated started at the end of a paragraph, and continued to the next, we selected both paragraphs. We continued selecting the lowest scoring snippet till the length of the summary became at least 500 characters long.

(2) The second difference is that users were allowed to view and modify their previous 10 judgments made through the CAL interface. Users may want to modify their previous judgments in case they make mistakes or if their understanding of relevance during assessments change.

2.2 User study system for the second submission

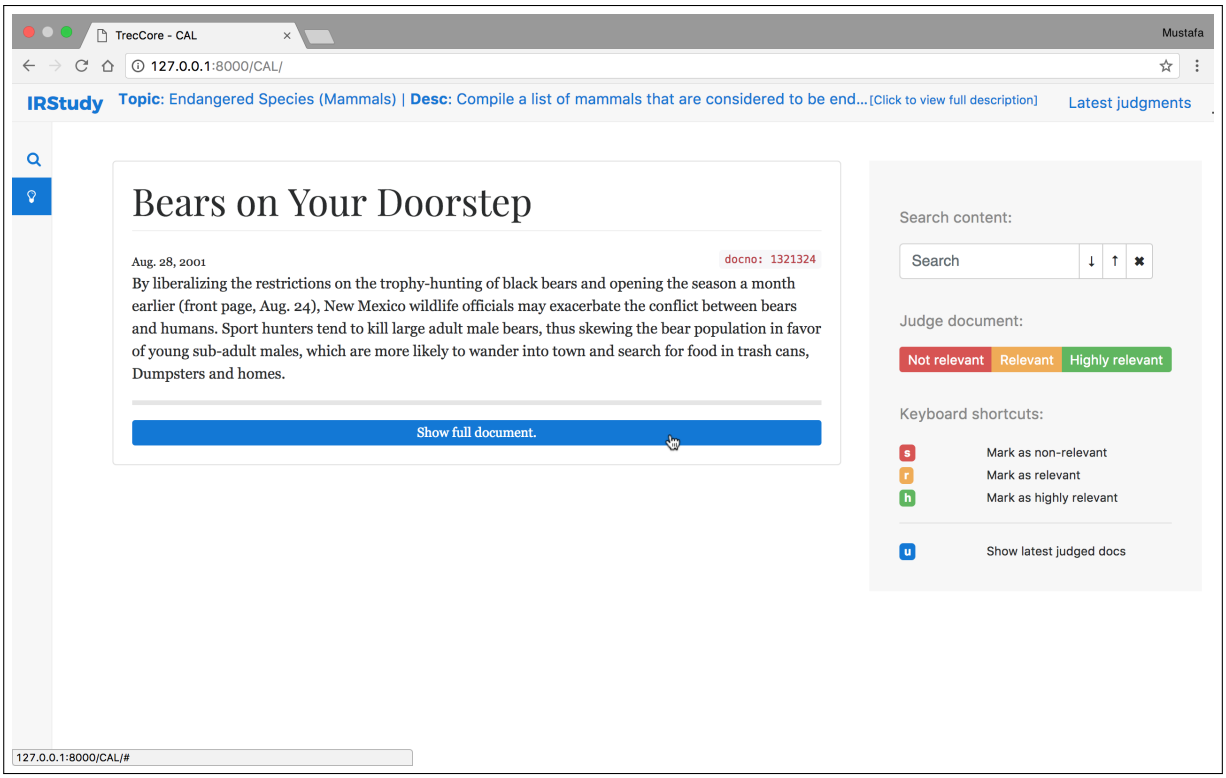

Fig. 4. CAL interface which shows paragraph with toggled full document in the user study system.

After we judged all the 250 topics using the prototype system described in Section 2.1, we improved our system based on our own experiences and feedback on the prototype system. The improved system was then applied to a real user study of 10 participants. The user study covered only 50 NIST topics. The judgments collected in the user study were then submitted as the second submission.

Changes made to the prototype system are described in the subsections below.

2.2.1 Modified Relevance Categories. We changed the relevance categories from {*Relevant*, *On topic* and *Not Relevant*} to {*Highly Relevant*, *Relevant* and *Not Relevant*}. We followed the same definition of relevance defined by Voorhees [16]: "Assume that you have the information need stated in the topic and that you are at home searching the web for appropriate material." and the same relevance categories:

- **Highly Relevant**: If the document directly addresses the core information need of the topic.
- **Relevant**: If the document contains information that you would find helpful in meeting your information need.
- **Not Relevant**: If the document is not relevant to the topic.

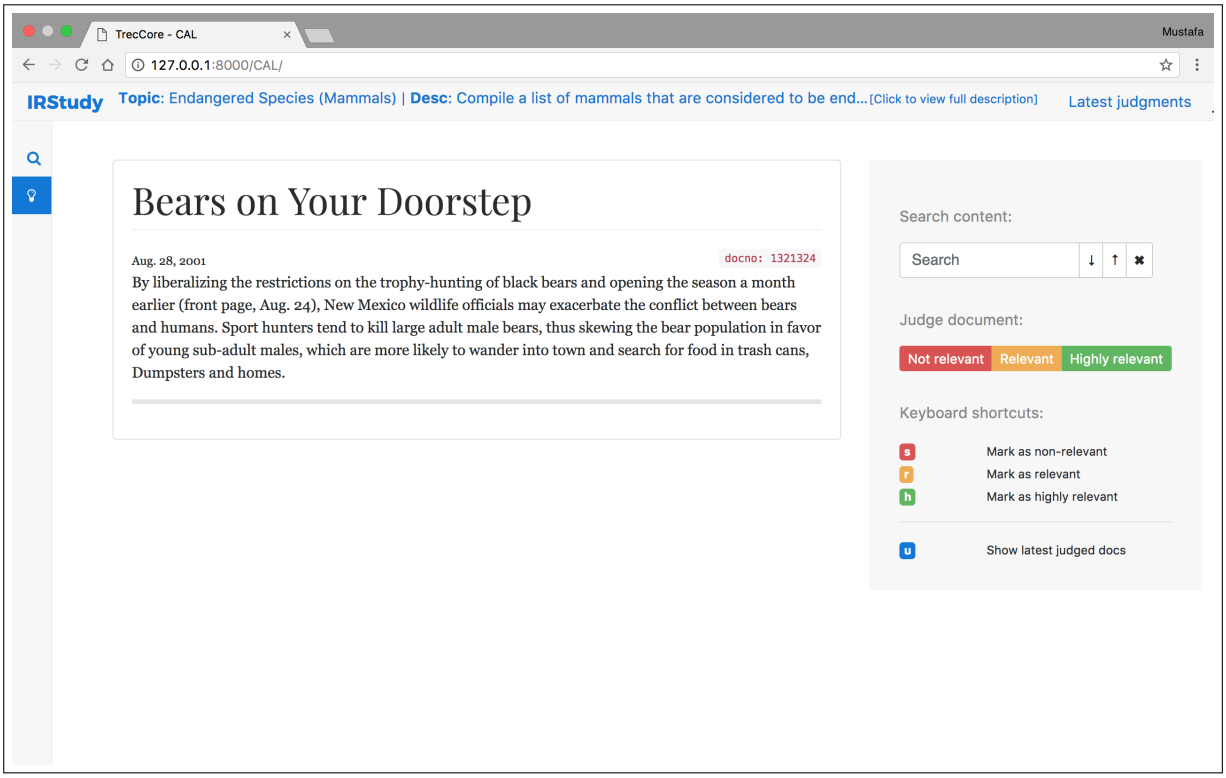

Fig. 5. CAL interface which only shows paragraph in the user study system.

2.2.2 Improved Search Options. We found the search model to be very helpful to find relevant documents during the beginning stage of assessment (when few relevant documents are available in the CAL model). Search was also very effective when the topic was hard (CAL model found only non-relevant documents). Instead of restricting users to 20 search results, we added an option to set the number of returned results (10, 20, 50, 100) in the search interface.

2.2.3 Improved CAL Model. For many topics in the prototype system, we realized the document summaries selected using the methods in Section 2.1.2 were inadequate for our information need (i.e. Figure 3). For some relevant documents, the displayed summary was not informative and lacked the necessary information to make a judgment. We therefore decided to replace the document summary generation algorithm with the summary selection method proposed in Zhang et al. [17] (i.e. Figure 5).

Instead of using a sentence as a summary of a document as proposed in Zhang et al. [17], we use paragraphs. A single paragraph usually contains more context than a sentence, while still being significantly shorter than the full document. The sentence feedback method proposed in Zhang et al. [17] was simulated on datasets from the TREC Total Recall track. Providing a single sentence to judge a document can be limiting, as it is often short in length and does not describe all aspects of the topic. However, paragraphs can provide more context and information than a single sentence.

We made several changes to our CAL system to adapt to paragraph relevance feedback. We first extracted and indexed all paragraphs from all the documents, and calculated the *tf-idf* features for each paragraph using the same document frequency (*df*) corresponding full documents (instead of counting each paragraph as a separate

document). The seed query, for which the initial classifier for each topic was built, was composed of the topic's title and description. The classifier is initially trained with 100 randomly chosen documents that we label as non-relevant. The classifier then ranks all documents paragraphs and select the highest ranking paragraph and its corresponding document. The highest ranking paragraph is then shown to the user for assessment.

We also provided the ability to view the full document content by clicking on the "View full document" button (Figure 4). Clicking the button would display the full document content below the paragraph.

Upon making a judgment, the relevance of the paragraph and its corresponding document were sent to the CAL model to retrain the classifier. Then, the next most likely paragraph with its corresponding document was selected from the remaining unassessed documents.

3 TWO STAGE ASSESSMENTS

As described in the Section 2, we designed two different systems for the two TREC Common Core deadlines. For the first deadline, six graduate students and one faculty member from the UWaterlooMDS team used the prototype system described in Section 2.1 to judge documents for all the 250 topics. For our own assessments, there was no time limit for single topic. The assessors could judge as many documents as they want and were free to use/switch between the search interface and the judging interface during the task.

We judged 31,661 unique documents for all the 250 topics and 6,854 unique documents for the 50 NIST topics. We also calculated the time taken for judging any document. This was done by calculating the difference between the timestamps at which the server received two successive judgments. In cases where judgments lasted more than 5 minutes, the judgment time for those documents were set to 5 minutes. We set the time spent to judge the first document of every topic to the average judgment time for that task. The average and median time for a single judgment over all 250 topics were 13 and 4.1 seconds respectively. The total time to judge all the 250 topics was 115.9 hours. For the 50 NIST topics, we spent 25.4 hours in total. An average of 27.6 minutes was spent on each topic.

For the second submission deadline, the improved user study system described in Section 2.2 was used to collect users' judgments in a controlled user study. We recruited 10 participants for the user study. We divided the 50 NIST topics into 10 sets, with each set having 5 different topics. Each participant judged 1 such set.

In order to compare the effectiveness and efficiency of different components of our interface, we designed 5 different treatments. These treatments are described in Table 1. The 5 treatments are described below:

- **Para**: Only the CAL model is active in this treatment. The CAL interface shows a single paragraph from the document for judgment, as shown in Figure 5.
- **Search/Para**: Both the search model and the CAL model are active in this treatment. The CAL interface shows a single paragraph.
- **Para&Doc**: Only the CAL model is active in this treatment. On the CAL interface, a single paragraph from the document is shown by default. However, users are able to view the full document content by clicking on the "View full document" button, as shown in Figrue 4.
- **Search/Para&Doc**: Both the search model and the CAL model are active for this treatment. The CAL interface shows a single paragraph but users can view the full document content if they wish.
- **Reference**: We randomly sampled 60 documents for each topic. The probability of each document being sampled is based on the relevance of that document. The interface shows the full documents one by one in a randomized order.

We wrapped our online user study system into a JavaScript executable application using Electron [5]. The purpose was to restrict participants from opening other applications/tabs while judging, and to allow us to better measure the time spent judging each document. Upon opening the application, a full-screen view of our system

[5]https://electron.atom.io/

Table 1. The 5 treatments used in the user study and their corresponding features.

Treatments	Search Model	CAL model showing Paragraph	CAL model showing toggle full document	Reference Model showing full document
Para	✗	✓	✗	✗
Search/Para	✓	✓	✗	✗
Para&Doc	✗	✓	✓	✗
Search/Para&Doc	✓	✓	✓	✗
Reference	✗	✗	✗	✓

is shown to the user. Users can exit the application by pressing the "*ESC*" keyboard button. Prior to judging, the participants completed a 1 hour in-lab tutorial on how to assess documents using the interface, relevance definition, user study procedure and so on. Participants were asked to install the application on their laptops and were instructed to complete the study at any time of the day but within 5 to 7 days. Each user had to finish 5 tasks, with each task having 1 topic and 1 treatment. We made sure each user was given 5 different topics and 5 different treatments. Moreover, we ensured that over 10 users, we covered all 50 topics.

For all tasks under the Para, Search/Para, Para&Doc and Search/Para&Doc treatments, users were allocated 1 hour of active time. Active time is the amount of time a user spends interacting with the system. Any mouse or keyboard action made within a 2 minute window contributes to the user's active time. As for the Reference treatment, users had to judge 60 documents without any time limit.

After analyzing the user study data, we found that the 10 users made a total of 8,093 judgments for all 50 topics.

4 COMPOSE SUBMITTED MANUAL RUNS

In total, we submitted 10 manual runs to TREC Common Core for the two submissions deadlines. The first 3 runs were composed using our own judgments from the first assessment stage (prototype system) and were submitted for the first deadline (June 18, 2017). For the remaining seven runs, they were based on a mix of our own judgments and the judgments collected from the user study in the second assessment stage. The seven runs were then submitted for the second deadline (July 31, 2017). Table 2 shows a brief description of each run.

4.1 First Submission

4.1.1 UWatMDS_TARSv1. The classifier was trained on positive examples ("Relevant" and "On Topic" documents) and negative examples ("Not Relevant" documents). The classifier then computed relevance score for each document. The unjudged documents were regarded as "Not Relevant". Therefore, the "Not Relevant" documents labeled by users were merged with unjudged documents.

For this run, every document was ranked using the tuple $< relevance, score >$. The documents were first ranked by "relevance" (where "Relevant" > "On-Topic" > "Not Relevant"), and then by the relevance scores. The top 10,000 ranked documents were selected to compose this run.

4.1.2 UWatMDS_TARSv2. This run is similar to UWatMDS_TARSv1 except for the sample weights used for training examples. A single training iteration in sofia-ml comprises of randomly sampling a positive and a negative example denoted by vectors **a** and **b**. The loss is computed over the difference of **a** and **b** [11]:

$$L(\mathbf{w}, \mathbf{a}, \mathbf{b}) = \frac{1}{1 + e^{<\mathbf{w}, \mathbf{a}-\mathbf{b}>}} \tag{4}$$

When using the sample weights s_a and s_b, we modified the loss function to

$$L(\mathbf{w}, \mathbf{a}, \mathbf{b}, s_a, s_b) = \frac{1}{1 + e^{<\mathbf{w}, |s_a|\mathbf{a} - |s_b|\mathbf{b}>}} \tag{5}$$

During classifier training for UWatMDS_TARSv1, "Relevant" and "On-Topic" documents were treated as a positive example ($weight = 1$), and "Non Relevant" documents were treated as a negative example ($weight = -1$). For this run, we experimented with various weights corresponding to each relevance label. Each parameter choice was evaluated using Mean Average Precision. We performed k-fold cross validation ($k = 5$) to select the best weight combination for each topic, which was used to train the final classifier for that topic. The run generation was performed in the same way as UWatMDS_TARSv1.

4.1.3 UWatMDS_HT10. This run used the same classifier used in UWatMDS_TARSv1. Documents were ranked by its relevance score. For this run, 10 documents of each topic were randomly sampled according to the probability estimated by the Horvitz–Thompson estimator [14, 18]. In our setting, the probability of each document being relevant was set to the reciprocal rank based on its classification score. The remaining 9990 documents in this run were selected based on their classification scores.

4.2 Second Submission

The second submission contained 7 runs. The judgments used for the second submission are described below:

Table 2. Brief description of the 10 submitted runs to TREC. Three sources of judgments were used to compose the runs. "G" judgments set were generated from Grossman and Cormack [6]. "M" judgments set were generated from the UWaterlooMDS team during the first assessment phase. "U" judgments set were generated from 10 participants of the user study during the second assessment phase.

Runs	Judgments			Topics	Precedence	Description
	G	M	U			
UWatMDS_TARSv1		✓		250	1	Ordered by {Rel, On-Topic, No-Rel} and each set was ranked by classification score.
UWatMDS_TARSv2		✓		250	3	Ordered by {Rel, On-Topic, No-Rel} and each set was ranked by tuned classification score.
UWatMDS_HT10		✓		250	2	Top 10 docs are sampled by HT Estimator and the rest are ranked by classification scores.
UWatMDS_ustudy			✓	NIST 50	1	Ordered by {Highly-Rel,Rel, No-Rel} and each set was ranked by classification score.
UWatMDS_AFuse	✓	✓		250	3	RRF Fusion of two rank lists from judgments in G, and M.
UWatMDS_AUnion	✓	✓		250	3	Ranked by classifier built on union of judgments in G, and M.
UWatMDS_AWgtd	✓	✓		250	3	Ranked by weighted classification score.
UWatMDS_BFuse	✓	✓	✓	NIST 50	2	RRF Fusion of three rank lists using judgments from G, M, and U.
UWatMDS_BUnion	✓	✓	✓	NIST 50	2	Ranked by classifier trained on union of judgments from G, M, and U.
UWatMDS_BWgtd	✓	✓	✓	NIST 50	2	Ranked by weighted classification score.

- **"G"**: is the relevance set generated from Grossman and Cormack [6]. Grossman and Cormack [6] used a variation of CAL and manually judged documents for 250 topics. They used binary relevance — relevant and non relevant to label each judged document.
- **"M"**: is the relevance set generated from UWaterlooMDS team in the first assessment stage.
- **"U"**: is built out of data collected using the five interface treatments from the user study experiment over the 50 NIST topics and 10 users.

UWatMDS_ustudy run was composed based on the judgments set "U" which was derived from real users' judgments. The six other runs investigated ways to use multiple judgments to produce a ranking. Three runs used the "G" + "M" relevance judgment sets (250 topics). The remaining runs used "G" + "M" + "U" (50 NIST). Judgment sets were merged using one of the three strategies:

- **Union**: A document is considered relevant if at least one relevance set marked it as relevant.
- **Weighted**: The judgment labels for documents are converted to integer labels: $0, 1, 2$ for "M" and "U", and $0, 1$ for "G". For any document, the new label is the sum of individual integer labels. The sample weights for each final integer label was determined using the same approach used in UWatMDS_TARSv2.
- **Fusion**: A classifier is trained on each judgment set (e.g. classifier for "G", classifier for "M" and classifier for "U") and each classifier is used to produce a rank list r. Every document d in the document set D has a rank $r(d)$ in rank list r. These rank lists R for different classifiers are fused into a final ranking using Reciprocal Rank Fusion [3]. The documents in the fused rank list are ordered based on their RRF_{score} as shown in Formula 6. In our experiment, k is set to 60.

$$RRF_{score}(d \in D) = \sum_{r \in R} \frac{1}{k + r(d)} \tag{6}$$

4.2.1 UWatMDS_ustudy. This run was built the same way as UWatMDS_TARSv1 but using the judgments "U" from user study. The classifier was trained on positive examples ("Highly Relevant" and "Relevant" documents) and negative examples ("Not Relevant"). The classifier computed classification score for each document. The unjudged documents were regarded as "Not Relevant".

For this run, every document was ranked using the tuple $< relevance, score >$ as the key (where "Highly Relevant" > "Relevant" > "Not Relevant"). The ranking strategy is the same as the run UWatMDS_TARSv1. The top 10,000 ranked documents were selected to compose this run.

4.2.2 UWatMDS_AFuse. Two different classifiers were built from the judgment "G" and "M" on 250 topics separately. Two ranked lists of documents were generated from these two classifiers and merged using reciprocal rank fusion.

4.2.3 UWatMDS_AUnion. Judgment sets "G" and "M" were merged using the **Union** strategy. For each topic, a classifier was trained using the merged judgment set and the resulting rank lists were used to compose the run.

4.2.4 UWatMDS_AWgtd. Judgment sets "G" and "M" were merged and classifiers were trained using the **Weighted** strategy.

4.2.5 UWatMDS_BFuse. Three different classifiers were built from the judgments set: "G" and "M" and "U" on 50 NIST topics separately. Three rank lists of documents were generated from these three classifiers separately. We fused these three rank lists using reciprocal rank fusion to compose this run.

4.2.6 UWatMDS_BUnion. We merge the three judgments sets "G", "M" and "U" using the union strategy. We built a classifier using the merged judgment set and the resulting rank list was used to compose the run.

4.2.7 UWatMDS_BWgtd. Judgment sets "G", "M" and "U" (50 NIST topics) were merged and classifiers were trained using the **Weighted** strategy.

Table 3. MAP, NDCG, P@10 and Recall@1000 for the 10 submitted runs evaluated on the NIST 50 topics only. The AP of each topic from every run is compared with the median AP from Manual Routing runs, Automatic Routing runs and Automatic Non-Routing runs, separately. The right side of the table shows the number of topics on each run with an AP greater than or equal to the median AP of the 3 different routing categories.

Runs	MAP	NDCG	P@10	Recall@1000	Contributed to the pools	# Topics with >= Median AP		
						Manual Ad Hoc	Auto Routing	Auto Ad hoc
UWatMDS_TARSv1	**0.46**	**0.70**	**0.83**	0.77	✓	39	**40**	45
UWatMDS_TARSv2	0.44	0.68	0.81	0.75	✗	32	38	43
UWatMDS_HT10	0.41	0.67	0.46	0.77	✗	28	36	39
UWatMDS_ustudy	0.32	0.57	0.65	0.68	✓	17	20	37
UWatMDS_AFuse	0.43	0.68	0.71	0.81	✗	42	37	**46**
UWatMDS_AUnion	0.40	0.65	0.65	0.77	✗	30	29	42
UWatMDS_AWgtd	0.40	0.66	0.68	0.79	✗	33	27	40
UWatMDS_BFuse	0.44	0.69	0.73	**0.82**	✓	**45**	38	**46**
UWatMDS_BUnion	0.38	0.64	0.64	0.77	✓	28	27	40
UWatMDS_BWgtd	0.42	0.66	0.74	0.78	✓	36	32	44

Table 4. Results of the different treatments from the user study over the NIST 50 topics. Every treatment covers 10 different topics, with no overlapping topics among treatments. The total number of judged documents, highly relevant, relevant and non-relevant documents judged by users using each treatment are shown. We also calculated the total number of True Positive (TP), averaged True Positive Rate (TPR) and averaged False Positive Rate (FPR) for each treatment based on the gold standard — NIST qrels. For all treatments except for the Reference, we only count judgments made from the start of a topic's task to the end of the first hour of active judging (inactivity time was not counted). Few users have spent more than 1 hour on some tasks but their judgments after the first hour were not counted. Tasks under the Reference treatment were not time-limited (users only needed to judge 60 sampled documents for the task's topic).

Treatments	#Highly-Rel by users	#Rel by users	#Non-Rel by users	#Highly-Rel & Rel by users	#Total judged by users	TPR vs. NIST	FPR vs. NIST	TP vs. NIST
Para	324	528	1410	**852**	2262	0.56	**0.43**	**453**
Search/Para	468	333	553	801	1354	0.72	0.62	403
Para/Doc	179	349	2168	528	**2696**	0.85	0.49	263
Search/Para+Doc	219	236	333	455	788	0.70	0.53	200
Reference	72	91	437	163	600	**0.86**	0.51	83

5 RESULTS AND DISCUSSION

We summarize our results in Table 3. Out of our 10 runs, only half of them contributed to the judgment pools. Among all the manual-non-routing runs, our runs had the highest average precision for 45 of the 50 topics. Among our runs, UWatMDS_TARSv1 achieved the highest MAP, NDCG and P@10 scores. It is worth mentioning that UWatMDS_TARSv1 was built only using the judgment set *M*. UWatMDS_BFuse achieves the highest recall@1000 among all our runs. As described in Section 4.2, UWatMDS_BFuse used the RRF fusion of rank lists obtained from three classifiers (each trained on judgment set "G", "M" and "U", respectively). Compared to UWatMDS_BUnion and UWatMDS_BWgtd which also used the same judgment sets, RRF fusion was able to build a run with higher MAP, NDCG, P@10 and recall. We also observe that UWatMDS_AFuse performs better than UWatMDS_AUnion and UWatMDS _AWgtd runs. Runs submitted to TREC Common Core for evaluation have three different categories depending on whether the runs used judgments from past tracks: **Manual Ad hoc** runs, **Automatic Routing**

runs and **Automatic Ad hoc**. Our runs fall under **Manual Ad hoc** since we did not use any judgments from past tracks. For each topic, we compare AP of our runs with the median AP of all the submitted runs. We observe that UWatMDS_BFuse and UWatMDS_TARSv1 runs perform relatively better than our other runs based on the number of topics with AP greater than or equal to the median.

We also show the results of different treatments from user study in Table 4. For each treatment, we calculated the number of judged documents across each relevance category. We observe that **Para** can help users find the more documents which are marked as *Highly Relevant* and *Relevant* by users within limited time. We evaluate the user study data using the gold standard - NIST qrels and measure the number of True Positives (documents considered relevant by the users and the NIST assessors). Tasks under the **Para** treatment retrieved the most number of relevant documents. In addition to TP, we also measure the True Positive Rate (TPR) and False Positive Rate (FPR). Tasks under the **Para** treatment have the lowest FPR.

6 ACKNOWLEDGMENT

Special thanks to Royal Sequeira and Salman Mohammed for their help with judgments during the first submission phase.

REFERENCES

[1] Gaurav Baruah, Haotian Zhang, Rakesh Guttikonda, Jimmy Lin, Mark D Smucker, and Olga Vechtomova. 2016. Optimizing Nugget Annotations with Active Learning. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 2359–2364.

[2] Stefan Büttcher, Charles LA Clarke, and Gordon V Cormack. 2016. *Information retrieval: Implementing and evaluating search engines*. Mit Press.

[3] Gordon V Cormack, Charles LA Clarke, and Stefan Buettcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. ACM, 758–759.

[4] Gordon V Cormack and Maura R Grossman. 2014. Evaluation of machine-learning protocols for technology-assisted review in electronic discovery. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 153–162.

[5] Gordon V. Cormack and Maura R. Grossman. 2015. Autonomy and Reliability of Continuous Active Learning for Technology-Assisted Review. *CoRR* abs/1504.06868 (2015). http://arxiv.org/abs/1504.06868

[6] Maura Grossman and Gordon Cormack. 2017. MRG_UWaterloo and WaterlooCormack Participation in the TREC 2017 Core Track: Notebook Draft. In *TREC*.

[7] Maura R. Grossman, Gordon V. Cormack, and Adam Roegiest. 2016. TREC 2016 Total Recall Track Overview. In *Proceedings of The Twenty-Fifth Text REtrieval Conference, TREC 2016, Gaithersburg, Maryland, USA, November 15-18, 2016*.

[8] Evangelos Kanoulas, Dan Li, Leif Azzopardi, and Rene Spijker. 2017. Clef 2017 technologically assisted reviews in empirical medicine overview. *Working Notes of CLEF* (2017), 11–14.

[9] Adam Roegiest, Gordon Cormack, Maura Grossman, and Charles Clarke. 2015. TREC 2015 Total Recall track overview. *Proc. TREC-2015* (2015).

[10] Mark Sanderson and Hideo Joho. 2004. Forming test collections with no system pooling. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 33–40.

[11] David Sculley. 2010. Combined regression and ranking. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 979–988.

[12] Mark D Smucker, James Allan, and Blagovest Dachev. 2012. Human question answering performance using an interactive document retrieval system. In *Proceedings of the 4th Information Interaction in Context Symposium*. ACM, 35–44.

[13] Trevor Strohman, Donald Metzler, Howard Turtle, and W Bruce Croft. 2005. Indri: A language model-based search engine for complex queries. In *Proceedings of the International Conference on Intelligent Analysis*, Vol. 2. Amherst, MA, USA, 2–6.

[14] Yves Tillé. 2006. *Sampling algorithms*. Springer Science & Business Media.

[15] Ellen M Voorhees. 2000. Variations in relevance judgments and the measurement of retrieval effectiveness. *Information processing & management* 36, 5 (2000), 697–716.

[16] Ellen M Voorhees. 2001. Evaluation by highly relevant documents. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 74–82.

[17] Haotian Zhang, Gordon V Cormack, Maura R Grossman, and Mark D Smucker. 2017. Evaluating Sentence-Level Relevance Feedback for High-Recall Information Retrieval. In *Submitting*.

[18] Haotian Zhang, Jimmy Lin, Gordon V Cormack, and Mark D Smucker. 2016. Sampling Strategies and Active Learning for Volume Estimation. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 981–984.

[19] Haotian Zhang, Wu Lin, Yipeng Wang, Charles L. A. Clarke, and Mark D. Smucker. 2015. WaterlooClarke: TREC 2015 Total Recall Track. In *Proceedings of The Twenty-Fourth Text REtrieval Conference, TREC 2015, Gaithersburg, Maryland, USA, November 17-20, 2015*.