A System for Efficient High-Recall Retrieval

Mustafa Abualsaud, Nimesh Ghelani, Haotian Zhang, Mark D. Smucker, Gordon V. Cormack, and Maura R. Grossman

University of Waterloo, Ontario, Canada
{m2abuals,nghelani,haotian.zhang,mark.smucker,gvcormac,maura.grossman}@uwaterloo.ca

ABSTRACT

The goal of high-recall information retrieval (HRIR) is to find all or nearly all relevant documents for a search topic. In this paper, we present the design of our system that affords efficient high-recall retrieval. HRIR systems commonly rely on iterative relevance feedback. Our system uses a state-of-the-art implementation of continuous active learning (CAL), and is designed to allow other feedback systems to be attached with little work. Our system allows users to judge documents as fast as possible with no perceptible interface lag. We also support the integration of a search engine for users who would like to interactively search and judge documents. In addition to detailing the design of our system, we report on user feedback collected as part of a 50 participants user study. While we have found that users find the most relevant documents when we restrict user interaction, a majority of participants prefer having flexibility in user interaction. Our work has implications on how to build effective assessment systems and what features of the system are believed to be useful by users.

CCS CONCEPTS

• **Information systems** → **Information retrieval**;

KEYWORDS

High-Recall; Electronic Discovery; Systematic Review

ACM Reference Format:
Mustafa Abualsaud, Nimesh Ghelani, Haotian Zhang, Mark D. Smucker, Gordon V. Cormack, and Maura R. Grossman. 2018. A System for Efficient High-Recall Retrieval. In *SIGIR '18: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, July 8–12, 2018, Ann Arbor, MI, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3209978.3210176

1 INTRODUCTION

The objective of high-recall information retrieval (HRIR) is to find all or nearly all relevant documents. Example applications of HRIR include electronic discovery (eDiscovery), systematic review and construction of information retrieval test collections.

Baseline Model Implementation (BMI) – a version of CAL (AutoTAR) – is the state-of-the-art approach to high-recall retrieval [2, 3, 5]. CAL uses a machine learning model with an iterative feedback process to retrieve documents. Our work builds on BMI.

SIGIR '18, July 8–12, 2018, Ann Arbor, MI, USA

ACM ISBN 978-1-4503-5657-2/18/07.
https://doi.org/10.1145/3209978.3210176

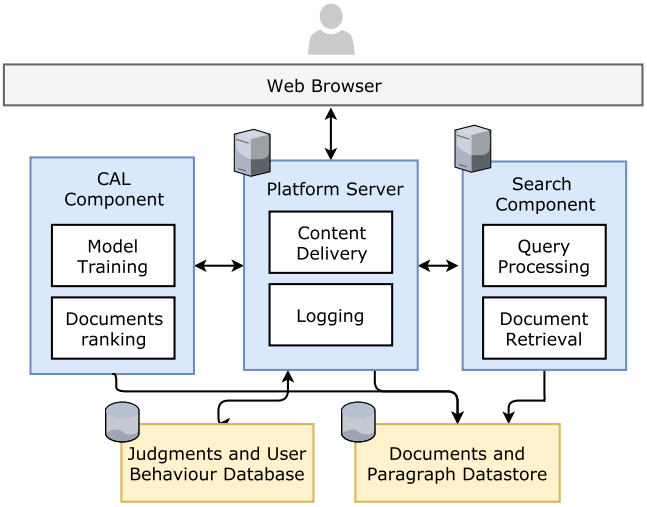

Figure 1: A high-level view of our platform architecture.

We demonstrate a platform for retrieving and assessing relevant documents that provides high data processing performance and a user-friendly document assessment interface. The platform is designed to increase assessors' assessment performance and thus reduce their review effort.

The performance of the platform was evaluated by 50 participants in a user experiment. In another work [10], we report on the design of user study and show preliminary results from the first 10 participants. In this paper, we show the data-processing performance of our platform, how it works, and how it is used and experienced by users.

2 SYSTEM ARCHITECTURE

Figure 1 shows the architecture of our platform. Our platform is composed of two different retrieval methods, CAL and Search. CAL automatically presents the user with documents based on a machine learning model, whereas Search allows users to enter queries and retrieve lists of documents. The platform server, as shown in the Figure 1, is responsible for displaying the document assessment interfaces to the user, configuring the system, and communicating requests and responses to and from each component. The types of user interactions that the server allows are:

- Set or change the topic of interest;
- Set seed query for building CAL's initial machine learning model;
- Search for specific documents using Search;
- Assess documents retrieved by either Search or CAL; and

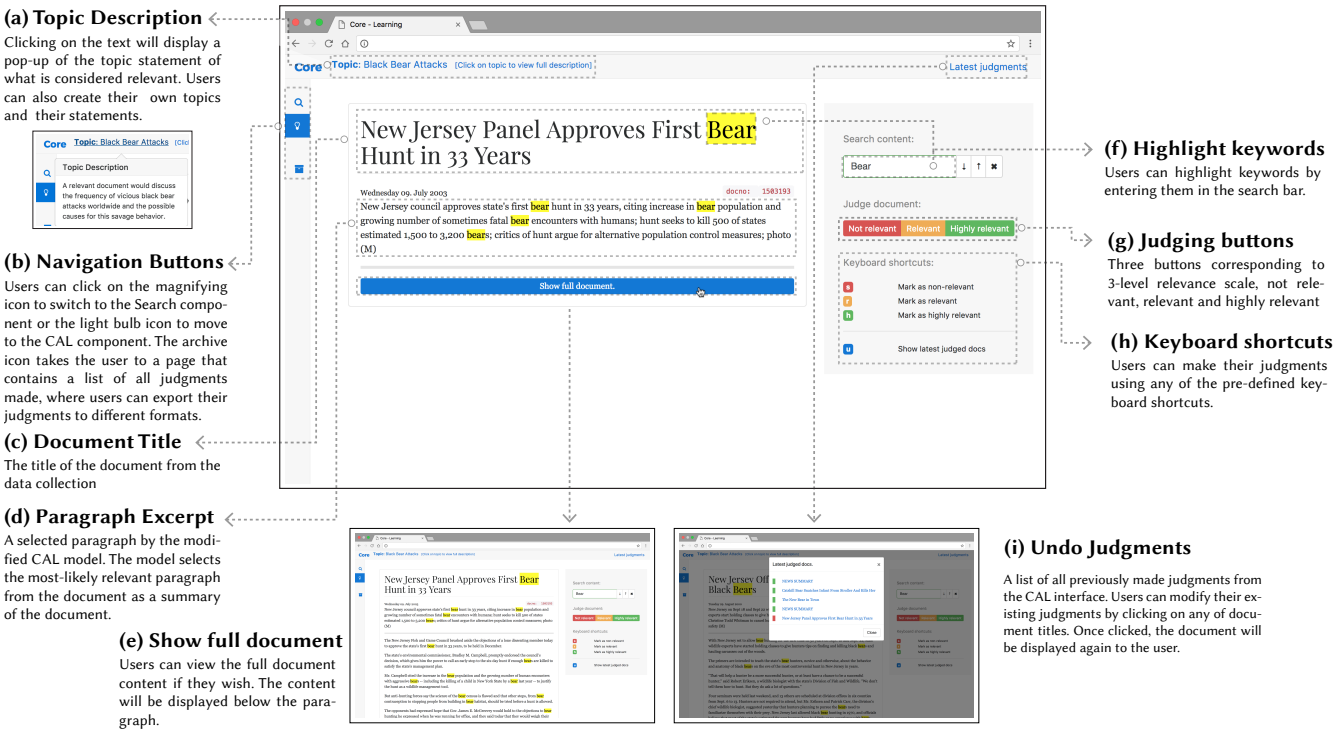

Figure 2: The continuous active learning (CAL) user interface.

- Export judgments sets.

All components in the architecture are stand-alone and interact with each other via HTTP API. For example, any search engine can be added to our system with minimal effort. This design also adds flexibility to these components and permits their use in different applications. For instance, the platform server was also used in the TREC 2017 RTS track evaluation for collecting judgments of tweets from the assessors [6]. The CAL component also has a command line interface through which various simulation experiments can be conducted.

The platform server is built using Django, a Python web framework. CAL and Search are written in C++. The source code is publicly available[1].

3 PLATFORM COMPONENTS & FEATURES

Our implementation of continuous active learning (CAL) is based on the Baseline Model Implementation (BMI)[2], which served as the baseline method in the TREC Total Recall Track 2015 and 2016 [4, 7]. BMI provides an iterative feedback process and uses a logistic regression classifier trained on relevance judgments from assessors. BMI uses this classifier to present the assessor with top scoring unjudged documents in the collection. After each iteration of judging and re-training, the learning model improves and returns the next most likely relevant documents to the user. We modified BMI as follows:

- **Paragraph-based:** A single paragraph is usually short and can contain enough material for a document to be assessed as relevant. Assessing the relevance of a document using a single paragraph can increase the total number of relevant documents found within a limited time and reduce review effort [10, 11]. Our CAL implementation retrieves the next likely relevant paragraph and presents it to the user for assessment, along with the option to view the full document content.
- **Frequency of model retraining:** The original algorithm processes relevance feedback in batches of size k, where k increases exponentially. Although having an increasing batch size reduces computation cost, it can delay the classifier from exploiting the potential benefits of newer relevance judgments. We made several improvements to the original algorithm to allow relevance feedback to be efficiently processed after each single judgment.

To meet the performance requirements of our modified algorithm, we implemented BMI in C++. In addition to using efficient data structures and parallel operations, we store all document and paragraph representations in memory. This enables quick training and scoring of documents/paragraphs. For the training of our logistic regression classifier, we used Sofia-ML[3]. We modified the Sofia-ML library to remove unnecessary features and the performance costs associated with them.

We compared our implementation by simulating the original BMI algorithm on the TREC 2017 Common Core test collection [1]. With around 1.8 million documents in the collection, the original

[1] https://cs.uwaterloo.ca/~m2abuals/code/SIGIR2018/
[2] http://cormack.uwaterloo.ca/trecvm/
[3] https://code.google.com/archive/p/sofia-ml/

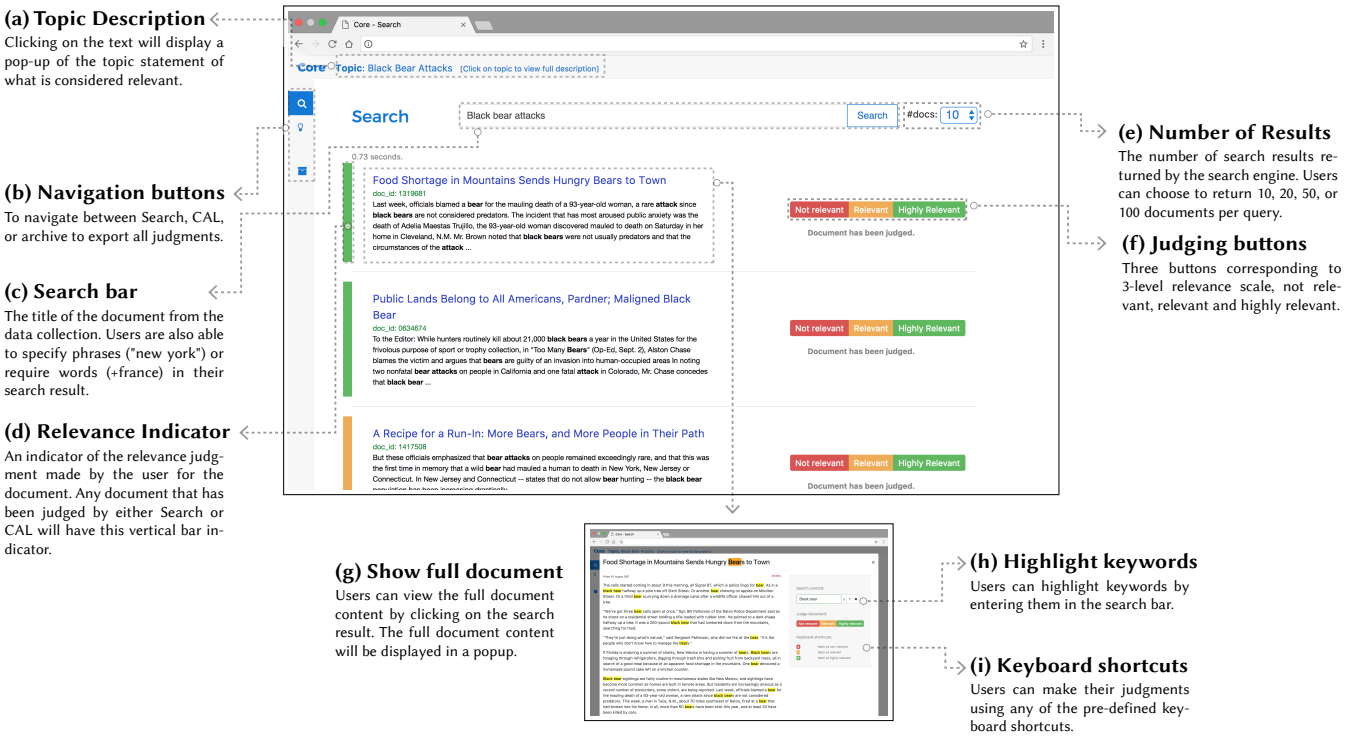

Figure 3: The search engine user interface.

implementation took an average of 294.3 seconds to train the model and score all the documents. In contrast, our implementation took an average of 3.5 seconds (1 thread) and 1.6 seconds (8 threads). It should be noted that the original implementation loads all the document features from disk every time the classifier is trained, while our implementation loads it once and keeps it in memory for subsequent use. This one-time loading cost in our implementation was 31.1 seconds, which is still significantly faster than the original implementation.

We also measured the performance of our implementation by simulating the modified algorithm on the same collection. The training and scoring of around 30 million paragraphs took an average of 2.1 seconds after every relevance judgment. Since this process causes a noticeable delay for users, we immediately present the next paragraph based on the previous scores while the new scores are being generated by the model. To reduce latency for the user, the browser front-end caches the top 10 highest-scoring paragraphs received from the CAL component. After a user makes a judgment, we immediately present the user with the next paragraph in the cache. The cache is flushed and updated every time it receives a new batch of paragraphs from the CAL component.

The modifications and the improvements we made increased the responsiveness of our system and removed any perceptible interface lag.

Figure 3 shows the interfaces of the Search component. We implemented the search component using Indri [9], but our platform is designed such that any search engine can be easily integrated. Our platform interacts with the search engine via HTTP API. Provided a search query, the platform expects the search engine to respond with a ranked list of documents with their summaries.

In order to help assessors find relevant documents easily, we included features that we found helpful as part of our own usage with a prototype version of our system. The features and their description are shown in Table 1. Screenshots of the CAL interface and features are shown in Figures 2 and 3.

4 USER STUDY DETAILS

We conducted a controlled user study to measure the performance our system. After receiving ethics approval from our university, we recruited 50 participants. The primary purpose of the study was to measure user performance with four variants of the system. A secondary purpose was to collect user feedback regarding the system and its features.

4.1 Corpus and Topics

We used the 50 NIST topics and the corpus provided by the TREC 2017 Common Core Track [1]. The corpus is The New York Times Annotated Corpus [8], which contains over 1.8 million news articles.

4.2 System Variants

The participants experienced all variations of our system during the study. For each variation, the participant spent 1 hour using

Table 1: A list of features participants rated.

Feature	Description	Example
Keyword Highlighting	Keyword search within a document or paragraph	Figure 2f, 3h
Judgment Shortcuts	Keyboard shortcuts for submitting relevance judgments	Figure 2h, 3i
Search Interface	Ability to use a search engine to find documents in addition to the learning interface	Figure 3
Topic Description	Display of topic statement of what is considered relevant	Figure 2a, 3a
Undo Judgments	Ability to review recent judgments and change judgment	Figure 2i
Full Document Content	Ability to view a full document rather than merely a paragraph summary	Figure 2e
Advance Search	For the search engine, the ability to specify phrases ("new york") or require words (+france)	Figure 3c

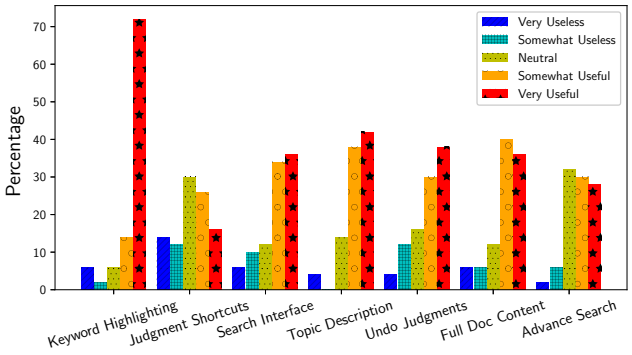

Figure 4: Percentage of user preference for different system features.

Table 2: Percentage of participants preferring a given system variant.

Treatments	Percentage of participants
CAL-P	16%
CAL-D	26%
CAL-P&Search	10%
CAL-D&Search	48%

our system to find as many relevant documents as they could for a given topic. The system variations were:

- **CAL-P**: CAL component with just paragraph summary. Search component is not enabled.
- **CAL-D**: CAL component with paragraph summary and option to view the full document. Search component is not enabled.
- **CAL-P&Search**: CAL-P with Search component enabled.
- **CAL-D&Search**: CAL-D with Search component enabled.

5 DISCUSSION

In this section, we compare the user performance and user feedback on different variants of our systems. At the conclusion of the study, we asked participants which system variant they preferred the most. 48% of study participants preferred CAL-D&Search over the more restrictive variants. Our participants want full control of a highly interactive system, but we found that performance is highest when their interactions are limited to producing relevance judgments on paragraph length excerpts.

Finally, we asked participants for their feedback on each of the system features in Table 1. We used a 5-point scale to rate each feature. The results are shown in Figure 4. The keyword highlighting feature is the most popular among all features, with 86% of users indicating that it was somewhat useful or very useful.

6 CONCLUSION

In this paper, we described the design of an efficient high-recall information retrieval system. The system allows for both iterative relevance feedback and search, and these components can be easily replaced with different implementations. We found that while participants preferred a system that gives them the flexibility to view full documents and interactively search, actual user performance was maximized when we limited interaction to submitting relevance judgments on paragraph length excerpts.

ACKNOWLEDGMENTS

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (Grants CRDPJ 468812-14, RGPIN-2017-04239, and RGPIN-2014-03642), in part by a Google Founders' Award, and in part by the University of Waterloo.

REFERENCES

[1] James Allan, Evangelos Kanoulas, Dan Li, Christophe Van Gysel, Donna Harman, and Ellen Voorhees. 2017. TREC 2017 Common Core Track Overview. In *TREC*.
[2] Gordon V Cormack and Maura R Grossman. 2014. Evaluation of machine-learning protocols for technology-assisted review in electronic discovery. In *SIGIR*.
[3] Gordon V. Cormack and Maura R. Grossman. 2015. Autonomy and Reliability of Continuous Active Learning for Technology-Assisted Review. *CoRR* abs/1504.06868 (2015).
[4] Maura R Grossman, Gordon V Cormack, and Adam Roegiest. 2016. TREC 2016 Total Recall Track Overview. In *TREC*.
[5] Maura R Grossman, Gordon V Cormack, and Adam Roegiest. 2017. Automatic and Semi-Automatic Document Selection for Technology-Assisted Review. In *SIGIR*.
[6] Jimmy Lin, Salman Mohammed, Royal Sequiera, Luchen Tan, Nimesh Ghelani, Mustafa Abualsaud, Richard McCreadie, Dmitrijs Milajevs, and Ellen Voorhees. 2017. Overview of the TREC 2017 Real-Time Summarization Track. In *TREC*.
[7] Adam Roegiest, Gordon V Cormack, Maura R Grossman, and C. L. A. Clarke. 2015. TREC 2015 Total Recall Track Overview. In *TREC*.
[8] Evan Sandhaus. 2008. The New York Times Annotated Corpus. (October 2008). LDC Catalog No.: LDC2008T19, https://catalog.ldc.upenn.edu/ldc2008t19.
[9] 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.
[10] Haotian Zhang, Mustafa Abualsaud, Nimesh Ghelani, Angshuman Ghosh, Mark D. Smucker, Gordon V. Cormack, and Maura R. Grossman. 2017. UWaterlooMDS at the TREC 2017 Common Core Track. In *TREC*.
[11] Haotian Zhang, Gordon V Cormack, Maura R Grossman, and Mark D Smucker. 2018. Evaluating Sentence-Level Relevance Feedback for High-Recall Information Retrieval. *arXiv* (2018).