



AFRL-RI-RS-TR-2018-115

DYNAMIC INFORMATION RETRIEVAL MODELING FOR DOMAIN-SPECIFIC SEARCH

GEORGETOWN UNIVERSITY

MAY 2018

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

**AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (<http://www.dtic.mil>).

AFRL-RI-RS-TR-2018-115 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /

FRANCESCA T. PAUGH
Work Unit Manager

/ S /

JON S. JONES
Technical Advisor, Information Intelligence
Systems and Analysis Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.					
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) MAY 2018		2. REPORT TYPE FINAL TECHNICAL REPORT		3. DATES COVERED (From - To) SEP 2014 – NOV 2017	
4. TITLE AND SUBTITLE DYNAMIC INFORMATION RETRIEVAL MODELING FOR DOMAIN-SPECIFIC SEARCH				5a. CONTRACT NUMBER FA8750-14-2-0226	
				5b. GRANT NUMBER N/A	
				5c. PROGRAM ELEMENT NUMBER 62702E	
6. AUTHOR(S) Grace Hui Yang				5d. PROJECT NUMBER MEMX	
				5e. TASK NUMBER 00	
				5f. WORK UNIT NUMBER 11	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Department of Computer Science, Georgetown University 37 th & OSTS NW Washington DC 20057				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/RIEA 525 Brooks Road Rome NY 13441-4505				10. SPONSOR/MONITOR'S ACRONYM(S) AFRL/RI	
				11. SPONSOR/MONITOR'S REPORT NUMBER AFRL-RI-RS-TR-2018-115	
12. DISTRIBUTION AVAILABILITY STATEMENT Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT Domain-specific search distinguishes itself from casual search in: (i) more complex and task-oriented information needs; (ii) use of a wider range of search strategies; (iii) requirement of a search session, rather than a one-shot query, to complete the complicated task; the search session often starts with a general query, follows by broadening or narrowing the search scope or drifting depending on the retrieved results, and continues this process until finding all aspects expected in the information need. These differences present new scientific challenges and call for more advanced domain-specific search algorithms. This research aims to create the next generation search engines, to be more specific, decision engines. The focus is on designing, experimenting, and deploying statistical models for modeling the dynamics presented in the search process.					
15. SUBJECT TERMS Information Retrieval, Reinforcement Learning, Dynamic Search, Domain-Specific Search					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT UU	18. NUMBER OF PAGES 118	19a. NAME OF RESPONSIBLE PERSON FRANCESCA T. PAUGH
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U			19b. TELEPHONE NUMBER (Include area code)

Contents

1	Summary	1
2	Introduction	3
2.1	Research Outline	4
2.2	Query Session Described	5
2.3	Dynamic Search Described	7
3	Methods, Assumptions, Procedures, Results and Discussion	8
3.1	Dynamic Search Modeling	12
3.1.1	Partially Observable Markov Decision Processes for Dynamic Search	12
3.1.2	Resolving the Efficiency Issue – Direct Policy Learning . . .	24
3.1.3	Modeling the Two-Way Communication	35
3.1.4	Bandits Approaches	44
3.1.5	Results and Discussion	48
3.2	Dynamic Search Evaluation	60
3.2.1	Text REtrieval Conference (TREC) Dynamic Domain Tracks 2015-2017	60
3.2.2	Annotating and Evaluating Complex Search Processes . . .	63
3.2.3	Finding Bounds for the Evaluation Metrics	65
3.2.4	Results and Discussion	72
3.3	Authorship Detection	77
3.3.1	Document Retrieval Phase	80
3.3.2	Document Clustering Phase	82
3.3.3	Results and Discussion	82
3.4	Innovative Search User Interface - Virtual Reality	86
3.4.1	Curved Text Display for Reading and Interacting with Documents	88
3.4.2	Keyword Queries	89
3.4.3	Structural Queries	90
3.4.4	Increasing Gesture Detection Accuracy	91
3.4.5	Results and Discussion	94
4	Tools and Deliverables	94
5	Conclusions	95
6	Appendix - Acronyms	102

List of Figures

1	Georgetown’s Accomplishments in the DARPA Memex Program (clockwise from top-left: book, DARPA evaluation results, innovative virtual reality user interface, and TREC Dynamic Domain Track task illustration).	2
2	A Markov chain of decision states in a search session. (S: decision states; q: queries; A: user actions such as query changes; D: documents).	9
3	Dual-agent stochastic game.	16
4	States.	16
5	Example History in Session Search (TREC 2014 Session 1011). . .	25
6	Influence Diagram for Session Search.	27
7	Interactions in Session Search. (Example was from TREC’14 Session 52. d_n was the n^{th} document in a ranked list D at iteration t .)	36
8	TREC 2011 Immediate Search Accuracy.	52
9	TREC 2012 Immediate Search Accuracy.	52
10	TREC 2013 Immediate Search Accuracy.	52
11	TREC 2014 Immediate Search Accuracy.	52
12	Long sessions (length greater than 4). Transition probabilities were listed with actions: Add (A), Remove (R), and Keep (K).	55
13	Factual and Specific sessions.	55
14	Factual and Amorphous sessions.	55
15	Intellectual and Specific sessions.	56
16	Intellectual and Amorphous sessions.	56
17	Memex Evaluation Results - Georgetown	59
18	Sample Topic of TREC 2017 DD Track.	60
19	Sample Relevance Judgement of TREC 2017 DD Track.	61
20	TREC Dynamic Domain (DD) Track Task Illustration.	61
21	Sample Feedback from the Simulated User (the Jig).	62
22	Upper bounds/ bound sizes on TREC16-DD topics	73
23	sDCG on TREC16-DD	75
24	CT on TREC16-DD	75
25	EU on TREC16-DD	76
26	Sample text from cybercrime dataset. Sensitive or offensive words were replaced by squares in this image.	78
27	Curved Text Display Interface	87
28	Virtual Keyboard	88
29	Grab a Keyword	89

30	AND query operator	91
31	Swipe	92
32	Fist Detection	93
33	Dumpling - A Dynamic Search Engine	95
34	Mineva	96
35	Mineva II	96
36	Mineva Visual	97
37	TREC Dynamic Domain Track Annotation Tool	97
38	TREC Dynamic Domain Track Simulated User (Jig)	98
39	Counter-Human-Trafficking Search Engine	99
40	Virtual Reality Search Engine User Interface	99

List of Tables

1	Examples in TREC Session Tracks	6
2	Symbols in the dual-agent stochastic game.	15
3	Example Features used in Direct Policy Learning.	33
4	Message Interpretations.	37
5	Featured used for Contextual Bandits	47
6	TREC 2011-2014 Session Track Dataset Statistics.	48
7	Search accuracy on TREC 2011 Session ($\dagger$ indicates a statistical significant improvement over Rocchio-CLK at $p < 0.05$ (t-test, one-sided)); * indicates a statistical significant improvement over QCM+DUP at $p < 0.05$ (t-test, one-sided)).	49
8	Search accuracy on TREC 2012 Session ($\dagger$ indicates a statistical significant improvement over Rocchio-CLK at $p < 0.05$ (t-test, one-sided)); * indicates a statistical significant improvement over QCM+DUP at $p < 0.05$ (t-test, one-sided)).	49
9	Search accuracy on TREC 2013 Session ($\dagger$ indicates a statistical significant improvement over Rocchio-CLK at $p < 0.01$ (t-test, one-sided)); * indicates a statistical significant improvement over QCM+DUP at $p < 0.01$ (t-test, one-sided)).	50
10	Search accuracy on TREC 2014 Session ($\dagger$ indicates a statistical significant improvement over Rocchio-CLK at $p < 0.01$ (t-test, one-sided)); * indicates a statistical significant improvement over QCM+DUP at $p < 0.01$ (t-test, one-sided)).	50
11	Search effectiveness of the Bandits approach from Iteration 1 to Iteration 10 on TREC DD 2016 Ebola Dataset (* indicates stat. significant improvement over rmit-lm-psg-max, ($p < 0.05$, t-test, one-sided))	56
12	A comparison of the scores of system with manual search completion and automated search completion (* indicates stat. significant improvement of automated system over manual system, ($p < 0.05$, t-test, one-sided)).	57
13	Example policy created by the bandit algorithm.	58
14	Document Relevance Scores in Ground Truth	67
15	The Importance of Normalization	67
16	Topic sample in TREC-DD 2016	72
17	Authorship Clustering Algorithm.	81
18	Experiments for Authorship Detection	85

1 Summary

Domain-specific search distinguishes itself from casual search in: (i) more complex and task-oriented information needs; (ii) use of a wider range of search strategies, including browsing (a.k.a. class search), author/inventor search, date search, citation search, backward/forward search, and ordinary keyword search; (iii) requirement of a search session, rather than a one-shot query, to complete the complicated task; the search session often starts with a general query, follows by broadening or narrowing the search scope or drifting depending on the retrieved results, and continues this process until finding all aspects expected in the information need. These differences present new scientific challenges and call for more advanced domain specific search algorithms.

In domain-specific search, users' search activities are no longer one-shot query – many queries are part of a more complex search task, which demonstrates multiple levels of complexity; search engines should appropriately incorporate itself into the bigger picture and help with the overall information seeking tasks which enables search engines to obtain clicks, mouse movements, eye tracking results, query reformulations and other forms of user feedback and engagement signals. Search engines are presented with the richest types of information and the biggest amount of data than ever and the complexity of the available information is tremendous. This demands search engines to upgrade from a retrieval system that basically looks for documents for single queries to a decision engine that picks the best choice for information seeking during the dynamic process.

The aim of this research was to create the next generation search engines, to be more specific, decision engines. The focus was on designing, experimenting, and deploying *statistical models* for modeling the dynamics presented in the search process. The technical challenges included:

- Given the complexity of the available data, how a search engine could integrate itself appropriately into the right places in the bigger context and serve for the ultimate tasks.
- Provide mathematical, theoretical, and practical support to formal modeling of user engagement and other dynamics in retrieval models for better retrieval effectiveness.
- Determine how to model user's exploration in the information space and optimize search engine's actions and algorithms to best help the user explore future states.
- Model interactions between user and search engine as well as interactions

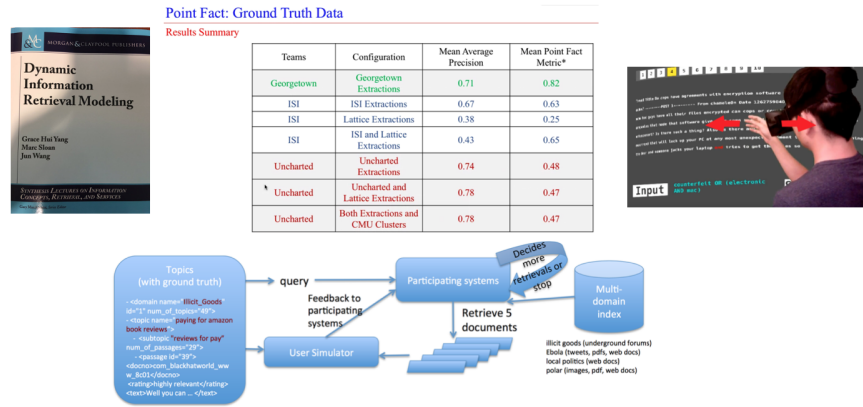


Figure 1: Georgetown's Accomplishments in the DARPA Memex Program (clockwise from top-left: book, DARPA evaluation results, innovative virtual reality user interface, and TREC Dynamic Domain Track task illustration).

among multiple users; create the dynamic environment for them to interact and to game with each other and achieve a win-win optimization.

- Model domain-specific search in its different stages and the learning curve present in the search process.

Starting in Sep 2014, the project successfully concluded in Nov 2017. The research team was strong in both publication and system development. The following achievements were accomplished by the PI and her team during the DARPA Memex Program (shown in Figure 1):

- Thirty publications in total, including **1 book on “dynamic information retrieval modeling”** [77], 4 journal papers on Association of Computing Machinery (ACM) Transaction of Information Systems (TOIS) [31, 81] and Information Retrieval Journal (IRJ) [30, 56], 13 conference papers in top venues including the ACM Special Interest Group on Information Retrieval (SIGIR), ACM Computer Human Interaction (CHI), ACM Web Search and Data Mining (WSDM), ACM International Conference on Theory of Information Retrieval (ICTIR), European Chapter of Information Retrieval (ECIR) conferences, and 12 workshop papers in ACM SIGIR, ACM CIKM, ACM WSDM, and the Text REtrieval Conference (TREC) conference workshops.
- Organization of academic evaluations with National Institute of Standards and Technology (NIST). The Georgetown team successfully led the Memex

effort in organizing evaluation campaign on dynamic and expert search at **TREC Dynamic Domain Tracks** from 2015 to 2017 [78, 79, 80]. Website <http://trec-dd.org/>.

- Highly-effective algorithms. The Georgetown team won **the best position** for finding answers for point-fact questions and also did very well for aggregate and clustering questions.
- Software delivered. During the three years working for the DARPA Memex project, the Georgetown team delivered software tools including Dumpling (a counter-human trafficking dynamic search engine), TREC Dynamic Domain (DD) Annotation Tool, and innovative dynamic search tool using virtual reality.

The Georgetown research team won **the best position** in DARPA Memex evaluation for finding answers of point-fact questions. The research funded 1 post-doc researcher, 1 programmer, 2 PhD students, and 7 Master's students. It graduated 4 Theses (2 PhDs, 1 Master's, and 1 undergrad). The research also won 1 best paper award (ACM ICTIR 2017 conference Best Student Paper Award).

The Georgetown Team made great impacts in the research community too. During this project, we organized 4 Workshops (TREC Dynamic Domain Track and SIGIR Privacy Track), with national and international participants from Japan, Canada, France, Austria, China, America, and UK. The PI also gave 4 tutorials on the funded research in top conferences (SIGIR 2014, WSDM 2015, ICTIR 2017, WSDM 2018). The Georgetown Team helped with releasing 3 Memex datasets (illicit goods, Ebola, local politics) to the public to enhance the research in domain-specific search. We also released a set of software tools as deliverables. They include Dumpling, Mineva, Mineva II, Minerva Visual, TREC DD Annotation, and CHT Search Engine.¹ Our experience in the DARPA Memex program has been truly rewarding.

2 Introduction

Dynamic search is an information retrieval task that involves a sequence of queries for a complex information need (e.g. searching for one-week tour plans in Italy). It is characterized by rich user-system interactions and temporal dependency between queries and between consecutive user behaviors. Dynamic search is a trial-and-error process, which matches well with the family of Reinforcement Learning (RL) algorithms: the RL algorithms learn from repeated, varied attempts which are

¹The publications and tools can be found at <http://infosense.cs.georgetown.edu/>

continued until success. The learner/agent learns from its dynamic interactions with the world. These similarities inspired us to model dynamic search using RL frameworks.

Dynamic search aims to find relevant documents for a session of multiple queries. It often happens when a user searches for an information need containing multiple aspects. For instance, the information need about *purchasing scooters* may compose of five sub information needs on *brands*, *store names*, *price information*, *quality* and *reliability*. A search log shows that a user issued 8 queries to accomplish the search task. Dynamic search also happens when an information need is vague or exploratory in nature. For example, for the information need about *planning a trip to the United States for a month with the constraint of only being able to travel within 150-mile radius of the destinations*, a user issued 21 queries in the session with various search attempts, such as checking the map and searching for the distances between two cities, to accomplish this task.

Despite its prevalence in daily life, before the work in this research, no effective dynamic search algorithm had been developed to try to solve this problem. Classical ad-hoc search algorithms treated each search iteration independently [62], which made them unsuitable for modeling dynamic search. It might seem convenient to use classic feedback models for dynamic search. However, these feedback models, such as Rocchio [37], used relevance judgements for no more than one search iteration. They were also not adequate in tackling dynamic search, either.

2.1 Research Outline

In this work, we proposed to model dynamic search as a dual-agent stochastic game, one of the standard variants of Partially Observable Markov Decision Processes (POMDPs), where the user agent and the search engine agent work together to jointly maximize their long-term rewards. In the framework, users' search statuses, such as exploitation, exploration, struggle etc. are modeled as hidden states, which can only be estimated through user interaction data. In each search iteration, one search algorithm is picked from a set of candidates to maximize a reward function. Our work provided a general framework to model dynamic search. It enabled the use of Reinforcement Learning algorithms for optimizing retrieval results. The experiments on the DARPA Memex Evaluation as well as on the Text REtrieval Conference (TREC) 2011–2014 Session Tracks, and the TREC Dynamic Domain (DD) 2015-2017 datasets showed a statistically significant improvement over the state-of-the-art dynamic search algorithms.

The design of states, actions, and rewards was quite flexible when using POMDPs in dynamic search. In the research, we also examined all available design choices from related work and compared their retrieval accuracy and efficiency. Our anal-

ysis revealed the effects of these choices, which enabled us to recommend practical design choices. The finding proved that modeling dynamic search using POMDPs is promising; however, it also showed that this approach is computational demanding.

To improve the efficiency of the above dynamic search models, we proposed another reinforcement learning framework, direct policy learning, which found optimal policies for the best search engine actions directly from what was observed in the user and search engine interactions via gradient descent. The proposed framework would greatly reduce the model complexity than the POMDP framework. It has a flexible design, which includes a wide range of features describing the rich interactions in dynamic search. The framework was shown to be highly effective and efficient on the experiments we accomplished and evaluations we participated in.

In this report, we present our research on modeling dynamic search for domain-specific search using reinforcement learning frameworks. We examined the characteristics of dynamic search and explained why reinforcement learning frameworks are more suitable for modeling dynamic search. We provided insights about major challenges when applying RL to model dynamic search and described our high-level solutions to these challenges. We also defined the scope of our research problem and highlighted the contributions of this work.

2.2 Query Session Described

A dynamic search session usually starts when a user writes a query according to his/her information need, submits the query to a search engine, and then receives a ranked list of documents from the search engine. After viewing the titles and snippets of documents in the list, the user may click on some documents and read them for a certain time period, which is called “*dwelt time*”. The user’s cognitive focus and understanding of the information need may change as the user gains more information from the documents. Oftentimes, the user reformulates a query and submits it to the search engine again. We termed the process from one query formulation to the next a “*search iteration*”. A session may last for several search iterations until the user stops the session either because of satisfaction of the information need [41] or abandonment [16].

Table 1 shows several examples of dynamic search from the Text REtrieval Conference (TREC) Session Tracks. For instance, in the TREC 2013 Session Track [41] session #9, the session proceeds as the following:

- Information Need *I*: “You want to know old US coins. Relevant information to you includes value of old US coins . . . how to start collecting old US coins ...”

Table 1: Examples in TREC Session Tracks

TREC’13 session 2, Information need: You want to buy a scooter. So you’re interested in learning more facts about scooters including: what brands of scooters are out there? What brands of scooters are reliable? Which scooters are cheap? Which stores sell scooters? which stores sell the best scooters?	
query1: scooter brands	query2: scooter brands reliable
query3: scooter	query4: scooter cheap
query5: scooter review	query6: scooter price
query7: scooter stores	query8: where to buy scooters
TREC’13 session 9, Information need: You want to know old US coins. Relevant information to you includes value of old US coins, types of old US coins, old US silver dollar, how to start collecting old US coins, how to sell old US coins and how to buy them, where to buy those coins.	
query1: old us coins	query2: collecting old us coins
query3: selling old us coins	query4: selling old “usa coins”
TREC’13 session 87, Information need: Suppose you’re planning a trip to the United States. You would be there for a month and able to travel within a 150-mile radius of your destination. With that constraint, what are the best cities to consider as possible destinations?	
query1: best us destinations	query2: distance new york boston
query3: maps.bing.com ...	query8: hartford visitors
query9: hartford connecticut tourism ...	
query19: philadelphia nyc travel	query20: philadelphia nyc train
query21: philadelphia nyc bus	
TREC’14 session 1011, Information need: ... You would like to provide your friend with relevant information about: different ways to quit smoking, programs available to help quit smoking, benefits of quitting smoking, second effects ...	
query 1: quit smoking	query 2: quit smoking hypnosis
query 3: side effects quit smoking	

- The user writes the first query q_1 = “old US coins”, and receives a ranked document set D1 from a search engine.
- The user clicks on some documents from D1. The clicked documents, along with clicking order and dwell time, are stored in C1. Through reading the documents, the user gains general information about old US coins.

- A new query $q_2 = \text{“collecting old US coins”}$ is submitted to the search engine for another sub information need in I . The session keeps going until the user ends the session.

2.3 Dynamic Search Described

Dynamic search demonstrates interesting properties that differ from ad-hoc search.

First there are *rich forms of interactions* when searching in a session. In a session, the user interacts frequently with the search engine by various forms of interactions, including query formulation, document clicks, document examination, eye movements, mouse movements, etc. These interactions provide abundant signals to describe the dynamic search process.

Second, there is *temporal dependency* when searching in a session. As we mentioned previously, a session consists of several search iterations. In early search iteration(s), the user learns from the retrieved documents, especially from those clicked documents that he/she has examined. This information influences the user's behavior, such as query formulation and document clicks, in the current search iteration. This temporal dependency between the current search iteration and the previous one(s) suggests the presence of the Markov property in dynamic search.

Third, it is a *trial-and-error* process when searching in a session. In a session, various forms of interactions are repeatedly carried out by the user iteration by iteration until the information need is satisfied [41] or the user is bored or frustrated [16]. The trial-and-error setting suggested that there is a reward mechanism which helps the user evaluate his/her attempts.

The unique challenges presented in dynamic search call for new statistical retrieval models. In addition, the inadequacy of the existing technologies motivated us to look for an ideal model for dynamic search. Such an ideal model should be able to:

- Model user interactions, which means it needs to contain parameters encoding actions;
- Model information need hiding behind user queries and other interactions;
- Set up a reward mechanism to guide the entire search algorithm to adjust its retrieval strategies; and
- Support Markov properties to handle the temporal dependency.

The family of RL algorithms [28] are designed to model trial-and-error settings: the algorithm learns from repeated, varied attempts which are continued until

success. The learner (also known as agent) learns from its dynamic interactions with the world, rather than from a labeled dataset as in supervised learning. In such a setting, a stochastic model assumes that the system’s current state depends on the previous state and action in a non-deterministic manner [57]. RL algorithms match very well with the above requirements.

In this research, we modeled dynamic search using reinforcement learning (RL) algorithms. Moreover, among various models in the RL family, we think Partially Observable Markov Decision Processes [70] is the most suitable model for dynamic search. In a POMDP, hidden information need can be modeled as hidden states, while visible interaction signals in the process can be modeled as observations or actions. In the following chapters of this report, we detail our methods, assumptions, procedures and results.

3 Methods, Assumptions, Procedures, Results and Discussion

We are often puzzled about what drives a user’s search in a session and why they make certain moves. We observe that sometimes the same user behavior, such as a drift from one subtopic to another, can be explained by opposite reasons: either the user is satisfied with the search results and moves to another sub information need, or the user is not satisfied with the search results and leaves the previous search path. The complexity of users’ decision making patterns makes dynamic search quite challenging [15, 76].

Researchers have attempted to find out the causes of topic drifting in dynamic search. The causes under study include personalization [76], task types [40, 53], and previous documents’ relevance [25]. However, the focus of this research was neither on identifying user intent nor on understanding the causes of such intent. Our work differs from existing work in that we believe a session goes through a series of hidden decision states, for which we designed a statistical retrieval model for dynamic search. Our emphasis was on effective retrieval model, not a user study about understanding user intent.

To model dynamic search using RL frameworks, we ran into several major challenges including the following. None of these challenges were trivial.

Challenge 1: How to design agents, states, actions and rewards in dynamic search? Dynamic search has attracted a great amount of research from a variety of approaches [14, 25, 47, 61, 86]. Before our research under the Memex program, researchers had just started to explore RL in Information Retrieval. There were only a few approaches at that time who used reinforcement learning

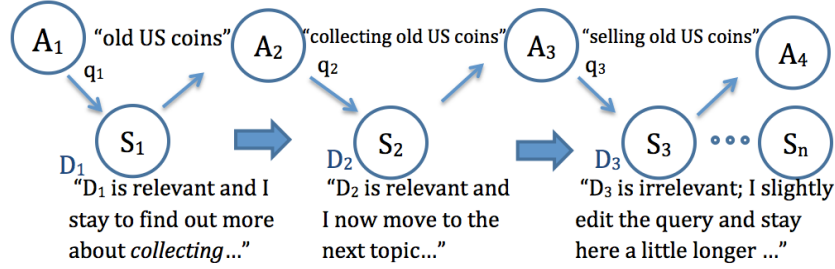


Figure 2: A Markov chain of decision states in a search session. (S: decision states; q: queries; A: user actions such as query changes; D: documents).

to study information retrieval problems. The approaches included using Markov Decision Process (MDP) [25], POMDP [36], exploratory online learning [28] and decision theories [66] to study interactive search. Although not all of them are based on POMDPs, they all share a similar idea of using states, actions, and rewards.

States, actions, and reward functions were the fundamental elements in a reinforcement learning framework. The following principles were usually referred to when defining these elements in a reinforcement learning framework:

- *States*: What changes with each time step?
- *Actions*: How does our system change the state?
- *Rewards*: How can we measure feedback or effectiveness?

In this research, we first modeled dynamic search as a dual-agent stochastic game, which was one variant of the POMDP algorithm. The two agents were the user agent and the search engine agent. The challenges of modeling dynamic search as a stochastic game lie in how to design and determine the decision states and actions of each agent, how to observe their behaviors, and how to measure the rewards and set the optimization goals. As a retrieval framework, we pay more attention to the search engine agent. When the search engine makes decisions, it picks a decision that jointly optimizes the common interest. In addition, for dynamic search, different design choices of states, actions and rewards might be available. Would different POMDP designs cause different retrieval performance? If yes, which design option was optimal? We addressed these questions in this work.

Challenge 2: How to solve the efficiency issue of RL frameworks in dynamic search?

Classic RL frameworks were computational demanding. In fact, most approaches often trade off model complexity for efficiency. Reducing the number of states and actions was often a requirement to achieve real-time interactions between the user and the search engine. For instance, only four decision-making states were modeled in our work.

However, even with a reduction of model complexity, these RL frameworks were still compute-intensive. We surveyed several RL frameworks on the TREC Session Track datasets to compare their retrieval effectiveness and efficiency, which included an MDP framework [25], a POMDP re-ranking framework [36], a decision theory framework [66] and the dynamic search framework. These frameworks all involved the design of states, actions and rewards. The survey results showed that although the dual-agent stochastic game model was the most effective framework among them, it was also one of the two slowest frameworks. The other one was the other POMDP model proposed by Jin et. al. [36]. Both of them took more than 9 hours to finish the tasks in the TREC Session Track. The MDP model [25] and the decision theory model [66] were much more efficient, which were able to finish in 2.5 hours. However, their retrieval accuracy significantly decreased. In the pursuit of efficiency, an obvious drawback of the prior work was that the details of the rich user-search-engine interactions could get lost. In this research, we further improved the efficiency and reduce the computational complexity of our dynamic search frameworks under the constraint of not losing much effectiveness.

Solution 1: To design the elements in our POMDP framework, we modeled the search engine and the user as two agents who work together to optimize a long-term goal.

The hidden decision states form a Markov chain in dynamic search. A Markov chain was a memoryless random process where the next state depends only on the current state [38]. Figure 2 illustrates a Markov chain of hidden decision states for TREC 2013 Session 9. In a session, a user’s judgment of the retrieved documents in the previous iteration affects or even decides the user’s actions in the next iteration. A user’s actions could include clicks, query changes, reading the documents, etc. The user’s gain, which we call *reward*, was the amount of relevant information that he or she obtains in the retrieved documents. The reward motivates the later user actions. If the decision states were known, we can use a MDP to model the process. However, in dynamic search, users’ decision states were hidden. We therefore model dynamic search as a POMDP [38].

In fact, not only the user, but also the search engine, makes decisions in a Markov process. A search engine takes in a user’s feedback and improves its retrieval algorithm iteration after iteration to achieve a better reward too. The search engine actions could include term weighting, turning on or turning off one or more of its search techniques, or adjusting parameters for the techniques. For

instance, based on the reward, the search engine can select p in deciding the top p documents used in pseudo relevance feedback [12].

Here dynamic search was modeled as a dual-agent stochastic game: the user agent and the search engine agent work together to maximize their long term rewards. The framework, which we termed “Win-Win search”, was based on POMDPs. In the Win-Win search framework, the decision states were hidden from our search engines’ retrieval algorithms and can only be inferred from user behavioral signals. The search engine actions were defined as a set of different retrieval algorithms, including relevance feedback [37], QCM [25], etc. The optimal retrieval algorithm was then picked to maximize an expected longterm reward function which returns a reward score based on the retrieval action to be taken and the current user decision state. We used different reward functions, such as nDCG@10 (normalized Discounted Cumulative Gain at cutoff 10 [40]) and user clicks, in the experiments.

However, the design of states, actions and rewards in Win-Win search were not the only choice when designing a RL framework for dynamic search. We examined related work and lay out the other available options for designing states, actions, and rewards in using POMDPs for dynamic search. We compared these design options from two aspects, retrieval accuracy and efficiency, and recommend practical design choices based on our observations. our analysis shows that POMDP frameworks, especially the Win-Win search, were indeed highly effective in modeling dynamic search. However, it also revealed that all these POMDP frameworks more or less suffered from low retrieval efficiency, which motivated the work in the direct policy learning research.

Solution 2: To improve the efficiency of the RL framework, we proposed a direct policy learning retrieval algorithm.

To further improve the effectiveness and efficiency of the dynamic search frameworks, we proposed a Direct Policy Learning (DPL) retrieval algorithm, where we aimed to create a framework that was able to incorporate a variety of features to adequately represent the complex interactions in a session. The new algorithm skipped the trouble of calculating beliefs for states, but directly learns the optimal policies by calculating mappings from observations to actions via gradient descent, and sequentially handles a search iteration in three phases, including a browse, a query, and a rank phase. During each iteration, the search engine maximized the long-term reward towards the ultimate information need. By learning over a history of actions and observations, the DPL algorithm was able to dramatically reduce computational complexity.

In addition to our originally proposed work, during the past three years working as part of the DARPA Memex program, we also worked on search engine evaluation (we were the main contributor to organize

the TREC Dynamic Domain Tracks from 2015 to 2017), authorship detection, question answering (and we won a top position in Memex evaluation), and innovative search engine user interface design and deployment.

In the remainder of this section, we detail our scientific and technical accomplishments in the following themes. They include dynamic search modeling for domain-specific search, improving the model efficiency, modeling the two-way communication between the user and the search engine, TREC Dynamic Domain 2015-2017 Organization efforts, authorship detection, and virtual reality user interface for dynamic search. The methods, results and discussions were organized into each corresponding sections.

3.1 Dynamic Search Modeling

3.1.1 Partially Observable Markov Decision Processes for Dynamic Search

Information Retrieval tasks were concerned with finding relevant documents to fulfill user's information needs. Dynamic search was an information retrieval task that involves multiple queries and multiple search iterations to achieve a complex information need [40, 41]. In a session, a user keeps formulating queries until he or she gets satisfied with the information need [41], bored, or frustrated [16]. This was a trial-and-error setting. Dynamic search was a challenging research area that was characterized by rich user-system interactions, complex information needs, and temporal dependency between queries and between user behaviors.

The family of Reinforcement Learning (RL) algorithms [28] matches well with the trial-and-error setting present in dynamic search: the algorithm learns from repeated, varied attempts which were continued until success. The learner (also known as agent) learns from its dynamic interactions with the world, rather than from a labeled dataset as in supervised learning. In such a setting, a stochastic model assumes that the system's current state depend on the previous state and action in a non-deterministic manner [57]. Among various models in the RL family, Partially Observable Markov Decision Processes [70] has been applied on IR problems including document re-ranking [36, 87] and advertisement bidding [84].

In this research, we proposed to model dynamic search as a *dual-agent stochastic game*. It was based on the POMDP framework. When there was more than one agent in a POMDP, the POMDP becomes a stochastic game (SG). The two agents in dynamic search were the user agent and the search engine agent. In contrast to most two-player scenarios such as chess games in game theory, the two agents in dynamic search were not opponents to each other; instead, they cooperate: they

share the decision states and work together to jointly maximize their goals. we we term the framework “**Win-Win search**” for its effort in ensuring that both agents arrive at a win-win situation. One may argue that in reality a commercial search engine and a user may have different goals and that was why some commercial search engines put their sponsors high in the returned results. However, this research focused on the win-win setting and assume a common interest – fulfilling the information needs – for both agents.

We evaluated the Win-Win search framework on TREC 2011 – 2014 Session Tracks. TREC 2011 - 2014 Session Tracks [39, 40, 41, 42] have spurred a great deal of research in dynamic search [14, 24, 25, 35, 53]. The tracks provide interaction data within a session and aim to retrieve relevant documents for the last query q_n in the session. The interaction data include queries, top returned documents, user clicks, and other relevant information such as dwell time. Document relevance was judged based on information need for the entire session, not just the last query.

The experiments showed that the proposed framework achieves statistically significant improvements over state-of-the-art interactive search and dynamic search algorithms.

Before we start to introduce our system, it was important to understand some preliminaries of reinforcement learning. The **agent** and the **environment** [74] were the two most common elements in an reinforcement learning framework. The *agent* was the learner in the learning framework, whose actions were aimed to be optimized. In dynamic search, there were actually two agents, the user and the search engine. *Environment* was the place where agents’ actions take place and where the consequences or rewards of actions were received from. In dynamic search, the environment can be the corpus. Beyond the agent and the environment, one can identify several other main elements: *states*, *actions*, *rewards*, *policies*, a *transition function*, and *value functions* [74].

- **States** S was a discrete set of states. In dynamic search, they can be queries [25] or hidden decision states.
- **Actions** A was a discrete set of actions that an agent can take. For instance, user actions include query changes, clicks, and reading the returned documents or snippets.
- **Reward** $r = R(s, a)$ was the immediate reward, also known as *reinforcement*. It gives the expected immediate reward of taking action a at state s .
- **Policy** π describes the behaviors of an agent. A non-stationary policy was a sequence of mapping from states to actions. π was usually optimized to

decide how to move around in the state space to optimize the long term reward $\sum_{t=1}^{\infty} r$.

- **Transition** T was the state transition function $T(s_i, a, s_j) = P(s_j|s_i, a)$. It was the probability of starting in state s_i , taking action a , and ending in state s_j . The sum over all actions gives the total state transition probability between s_i and s_j $T(s_i, s_j) = P(s_j|s_i)$; which was similar to the state transition probability in the Hidden Markov Model (HMM) [8].
- **Value function and Q-function** Given a policy π at time t , a value function V calculates the expected long term reward starting from state s inductively: $V_{\pi,t}^*(s) = R(s, \pi_t(s)) + \gamma \sum_{s'} T(s, a = \pi_t(s), s') V_{\pi,t+1}^*(s')$, where the initial value $V_{\pi,t=1}(s) = R(s, a = \pi_{t=1}(s))$, s was the current state, s' was the next state, $a = \pi_t(s)$ was any valid action for s at t , γ was a future discount factor. Usually an auxiliary function, called the **Q-function**, was used for a pair of (s, a) : $Q^*(s_t, a) = R(s_t, a) + \gamma \sum_a P(s_t|s_{t+1}, a) \max_a Q(s_{t+1}, a)$, where $R(s_t, a)$ was the immediate reward at t , a was any valid action at $t + 1$. Note that $V^*(s) = \max_a Q^*(s, a)$.
- **Markov Decision Process (MDP)** provides the basics for the Win-Win search framework. An MDP was composed by agents, states, actions, reward, policy, and transitions [38]. An agent takes inputs from the environment and outputs actions. The actions in turn influences the states of the environment. An MDP can be represented by a tuple $\langle S, A, T, R \rangle$.
- **Q-Learning** Reinforcement learning algorithms provides solutions to MDPs [38]. The most influential RL algorithm was Q-learning. Given a Q-function and a starting state s , the solution can be a greedy policy that at each step, it takes the action that maximizes the Q-function: $\pi^*(s) = \arg \max_a [R(s, a) + \gamma \sum_a T(s, a, s') Q(s', a)]$, where the base case maximizes $R(s_1, a)$.
- **Partially Observable MDP (POMDP)** When states were unknown and can only be guessed through a probabilistic distribution, an MDP becomes a POMDP [38]. POMDP was represented by a tuple $\langle S, A, T, \Omega, O, B, R \rangle$, where S, A, R were the same as in MDP. Since the states were unknown, the transition function T models transitions between beliefs, not transitions between states any more: $T : B \times A \times B \rightarrow [0, 1]$.
- **Belief** B was the set of beliefs defined over S , which indicates the probability that an agent was at a state s . It was also known as *belief state*.

Table 2: Symbols in the dual-agent stochastic game.

Name	Symbol	Meanings
State	S	the four hidden decision states in Figure 4
User action	A_u	add, remove and keep query terms
Search engine action	A_{se}	increase/decrease/keep term weights, adjust search techniques, etc
Message from user to search engine	Σ_u	clicked and SAT clicked documents
Message from search engine to user	Σ_{se}	top k returned documents
User’s observation	Ω_u	observations that the user makes from the world
Search engine’s observation	Ω_{se}	observations that the search engine makes from the world and from the user
User reward	R_u	relevant information the user gains from reading the documents
Search engine reward	R_{se}	nDCG that the search gains by returning documents
Belief state	B	belief states generated from the belief updater and shared by both agents

- **Observations** Ω was a discrete set of observations that an agent makes about the states. O was the observation function which represents a probabilistic distribution for making observation ω given action a and landing in the next state s' . A major difference between an HMM and a POMDP was that POMDP considers actions and rewards while HMM does not.

A session can be viewed as a Markov chain of evolving states. Every time when a new query was issued, both the user and the search engine transition into a new state. In our setting, the two agents work together to achieve a win-win goal.

Figure 3 shows the proposed dual-agent SG, which was represented as a tuple $\langle S, A_u, A_{se}, \Sigma_u, \Sigma_{se}, \Omega_u, \Omega_{se}, O, B, T, R \rangle$. The symbols are defined in Table 2.

- S was the decision states.
- A_u, A_{se}, Σ_u , and Σ_{se} were the actions. we divide the actions into two types: domain-level actions A , what an agent acts on the world directly, and communication-level actions Σ , also known as *messages*, which only go between the agents. User actions A_u were mainly query changes [25] while search engine actions A_{se} were term weighting schemes and adjustments to search techniques. Both Σ_u and Σ_{se} were sets of relevant documents, that an agent uses to inform the other agent about what they consider as relevant.
- Ω was the observation that an agent can draw from the world or from the other agent. O was the observation function that maps states and actions

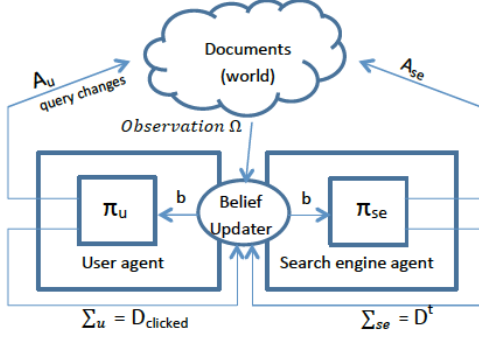


Figure 3: Dual-agent stochastic game.

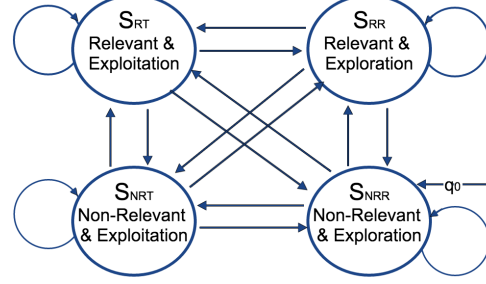


Figure 4: States.

to observations: $O : S \times A \rightarrow \Omega$ or $O : S \times \Sigma \rightarrow \Omega$. Note that the actions can be domain-level actions A or messages Σ , or a combination of both.

- B was the set of belief states that were shared by both agents. The beliefs were updated every time when an observation happens. There were two types of belief: $B_{\bullet\Sigma}$, beliefs before the messages and $B_{\Sigma\bullet}$, beliefs after the messages.
- The reward function R was defined as $B \times A \rightarrow \mathbb{R}$. It was the amount of document relevance that an agent obtains from the world. R_{se} was the nDCG score [40] that the search engine gains for the documents it returns. R_u was the relevant information that the user gains from reading the documents. Our retrieval algorithm jointly optimized both R_u and R_{se} .

The following describes their interactions in the stochastic game:

1. At search iteration $t = 0$, both agents begin in the same initial state S^0 .
2. t increases by one: $t = t + 1$. The user agent writes a query q^t and takes the t^{th} user agent action a_u^t , which was a query change from the previous query.
3. (*) The search engine agent makes observations Ω_{se} from the world and updates its before-message-belief-state $b_{\bullet\Sigma_{se}}^t$ based on $O(S^t, a_u^t, \Omega_{se})$.
4. The search engine runs its optimization algorithm and picks the best policy π_{se} , which maximizes the joint long term rewards for both agents. Following the policy, it takes actions a_{se}^t . This was where the model performs retrieval.

5. Search engine action a_{se}^t results in a set of documents D^t , which were returned as message Σ_{se}^t sent from the search engine agent to the user agent through the world.
6. (*) The user agent receives message Σ_{se}^t and observes Ω_u . If the user would like to stop the search, the process ends. Otherwise, the user updates the after-message-belief-state $b_{\Sigma_{\bullet}^t u}^t$ based on $O(S^t, \Sigma_{se}^t, \Omega_u)$.
7. Based on the current beliefs, the user agent sends its feedback messages Σ_u^t to inform the search engine agent. Σ_u^t were clicks, some of which were SAT clicks. It contains a set of documents $D_{clicked}$.
8. (*) The search engine agent observes Ω_{se} from the world and updates its after-message-belief-state $b_{\Sigma_{\bullet}^t se}^t$ based on $O(S^t, \Sigma_u^t, \Omega_{se})$.
9. The user agent picks a policy π_u , which we don't study here, and continues to send out actions a_u^{t+1} in the form of query changes. The world moves into a new state s_{t+1} . $t = t + 1$. The process repeats from step 3.

Steps 3, 6, and 8 happen after making an observation from the world or from the other agent. They then all involve a belief update.

We often observe that the same user behavior in dynamic search may be motivated by different reasons. For instance, in TREC 2013 Session 2, a user searches for “scooter brands” as q_1 and finds that the 6th returned document with title “*Scooter Brands - The Scooter Review - The Scooter Review*” was relevant. The user clicks this document and reads it for 48 seconds, which we identify as a SAT click since it lasts more than 30 seconds [26]. Next, the user adds a new term ‘reliable’ into q_1 to get q_2 = “scooter brands reliable”. We notice that ‘reliability’ does not appear in any previously retrieved documents D_1 . It suggests that the user was inspired to add ‘reliability’ from somewhere else, such as from personal background knowledge or the information need. In this case, she finds *relevant* documents from the previously retrieved documents but still decides to *explore* other aspects about the search target.

On the contrary, in TREC 2013 Session 9 q_1 , another user searches for “old US coins” and also finds relevant documents, such as a document about “... *We buy collectible U.S.A. coins for our existing coin collector clients...*”. He adds a new term ‘collecting’ to get the next query q_2 “collecting old us coins”. After reducing the query terms and document terms into their stemmed forms, the added term ‘collecting’ does appear in this document as we can see. It suggests that the user selects a term from the retrieval results and hones into the specifics. In this case, he finds *relevant* documents in the previously retrieved documents and decides to *exploit* the same sub information need and investigates it more.

We observe that even if both users show the same search behavior, e.g. adding terms, the reasons vary: one was adding the new search term because the original search results were not satisfactory, while the other was because the user wants to look into more specifics. This made us realize that document relevance and users' desire to explore were two independent dimensions in deciding how to form the next query.

Inspired by earlier research on user intent and task types [53, 75] and our own observations, we proposed four hidden decision making states for dynamic search. They were identified based on two dimensions: 1) "relevant dimension" – whether the user thinks the returned documents were relevant, and 2) "exploration dimension" – whether the user would like to explore another subtopic. The two dimensions greatly simplify the complexity of user modeling in dynamic search. The relatively small number of discrete states enabled us to proceed with POMDP and its optimization at low cost.

The cross-product of the two dimensions result in four states: i) user finds a relevant document from the returned documents and decides to explore the next sub information need (relevant and exploration, e.g. *scooter price* $\rightarrow$ *scooter stores*), ii) user finds relevant information and decides to stay in the current sub information need to look into more relevant information (relevant and exploitation, e.g. *hartford visitors* $\rightarrow$ *hartford connecticut tourism*), iii) user finds out that the returned documents were not relevant and decides to stay and try out different expressions for the same sub information need (non-relevant and exploitation, e.g. *philadelphia nyc travel* $\rightarrow$ *philadelphia nyc train*), iv) user finds out that documents were not relevant and decides to give up and move on to another sub information need (non-relevant and exploration, e.g. *distance new york boston* $\rightarrow$ *maps.bing.com*).

Figure 4 shows the decision state diagram for Win-Win search. The subscriptions stand for $\{RT = \text{RelevantexploitTation}, RR = \text{RelevantexploRation}, NRT = \text{NonRelevantexploitTation}, NRR = \text{NonRelevantexploRation}\}$. We insert a dummy starting query q_0 before any real query and it always goes to S_{NRR} . The series of search iterations in a session move in the decision states from one to the next. A sequence of states can be time stamped and presented as $s_t = S_m$, where $t = 1, 2, \dots, n$ and $m = \{RT, RR, NRT, NRR\}$.

There were two types of actions in the framework, domain-level actions and communications-level actions.

The domain-level actions A_u and A_{se} represent the actions directly performed on the world (document collection) by the user agent and by the search engine agent, respectively.

The common user actions include *writing a query*, *clicking a document*, *SAT clicking a document*, *reading a snippet*, *reading a document*, *changing a query*, and *eye-tracking the documents*. In this research, we only study query changes and

clicks as user actions. However, the framework can be easily adopted for other types of user actions.

Query changes Δq [25] consist of *added query terms* $+\Delta q_t = q_t \setminus q_{t-1}$, *removed query terms* $-\Delta q_t = q_{t-1} \setminus q_t$, and *theme terms* $q_{theme} = LongestCommonSubsequence(q_t, q_{t-1})$. For example, in Session 87, given q_{19} = “philadelphia nyc travel” and q_{20} = “philadelphia nyc train”, we obtain the following query changes: $q_{theme} = LCS(q_{19}, q_{20}) = \text{“philadelphia”}$, $-\Delta q_{20} = \text{“travel”}$, and $+\Delta q_{20} = \text{“train”}$. All stopwords and function words were removed.

The search engine domain-level actions A_{se} include increasing, decreasing, and maintaining the term weights, as well as adjusting parameters in one or more search techniques.

The second type of actions were communication-level actions (messages) Σ_u and Σ_{se} . They were actions that only performed between agents.

In our framework, the messages were essentially *documents* that an agent thinks were relevant. Σ_u was the set of documents that the user sends out; we define them as the clicked documents $D_{clicked}$. In TREC 2013 Session, 31% search iterations contain SAT clicked documents. 23.9% sessions contain 1 to 4 SAT clicked documents, and a few sessions, for instance Sessions 45, 57 and 72, contain around 10 SAT clicked documents. 88.7% SAT clicked documents appear in the top 10 retrieved results.

Similarly, Σ_{se} was the set of documents that the search engine sends out. They were the top k returned documents (k ranges from 0 to 55 in the TREC setting). They demonstrate what documents the search engine thinks were the most relevant. In TREC 2013, 2.8% (10) search iterations return less than 10 documents, 90.7% (322) return exactly 10, 5.1% (18) return 10~20, and 1.4% (5) return 20~55 documents.

The observation function $O(s_j, a_t, \omega_t)$, defined as $P(\omega_t | s_j, a_t)$, was the probability of observing $\omega_t \in \Omega$ when agents take action a_t and land on state s_j . To simplify the calculation, we assume that the observation of *relevance* and *exploration* were independent. The first type of observation was related to *relevance*. In Step 8, after the user sends the message Σ_u (user clicks) out at Step 7, the search engine updates its after-message-belief-state $b_{\Sigma_{\bullet se}}$ based on its observation of user clicks. The observation function for ‘Relevant’ states was :

$$O(s_t = \text{Rel}, \Sigma_u, \omega_t = \text{Rel}) \stackrel{def}{=} P(\omega_t = \text{Rel} | s_t = \text{Rel}, \Sigma_u) \quad (1)$$

It can be written as $\frac{P(\omega_t = \text{Rel}, s_t = \text{Rel}, \Sigma_u)}{P(s_t = \text{Rel}, \Sigma_u)}$. By taking $P(s_t = \text{Rel}, \Sigma_u)$ as a constant, we can approximate it by $P(\omega_t = \text{Rel}, s_t = \text{Rel}, \Sigma_u) = P(s_t = \text{Rel} | \omega_t = \text{Rel}, \Sigma_u)P(\omega_t = \text{Rel}, \Sigma_u)$. Given that user clicks Σ_u were highly correlated to ω_t , we can approximate $P(s_t = \text{Rel} | \omega_t = \text{Rel}, \Sigma_u)$ by $P(s_t = \text{Rel} | \omega_t = \text{Rel})$. Further,

by taking $P(\Sigma)$ as a constant, we have

$$\begin{aligned} O(s_t=\text{Rel}, \Sigma_u, \omega_t=\text{Rel}) &\propto P(s_t = \text{Rel}|\omega_t = \text{Rel})P(\omega_t = \text{Rel}, \Sigma_u) \\ &\propto P(s_t = \text{Rel}|\omega_t = \text{Rel})P(\omega_t = \text{Rel}|\Sigma_u) \end{aligned} \quad (2)$$

Similarly, we have

$$\begin{aligned} O(s_t=\text{Non-Rel}, \Sigma_u, \omega_t=\text{Non-Rel}) \\ \propto P(s_t = \text{Non-Rel}|\omega_t = \text{Non-Rel})P(\omega_t = \text{Non-Rel}|\Sigma_u) \end{aligned} \quad (3)$$

as well as $O(s_t=\text{Non-Rel}, \Sigma_u, \omega_t=\text{Rel})$ and $O(s_t=\text{Rel}, \Sigma_u, \omega_t=\text{Non-Rel})$.

Based on whether a SAT Click exists or not, we calculate the probability of the SG landing at the “Relevant” states or the “Non-Relevant” states (the first dimension of hidden decision states). A SAT Click means that a user clicked a webpage and dwell on that page for more than 30 seconds. In a lot of literature, SAT Clicks were used to infer that users find relevant content and have spent enough time to examine and digest the content. Hence at search iteration t , if the set of previously returned documents leads to one or more SAT clicks, the current state was likely to be relevant, otherwise non-relevant. That was to say,

$$s_t \text{ was likely to be } \begin{cases} \text{Relevant} & \text{if } \exists d \in D_{t-1} \text{ and} \\ & d \text{ was SATClicked} \\ \text{Non-Relevant} & \text{otherwise.} \end{cases}$$

Based on this intuition, we calculate $P(\omega_t = \text{Rel}|\Sigma_u)$ and $P(\omega_t = \text{Non-Rel}|\Sigma_u)$ as:

$$P(\omega_t = \text{Rel}|\Sigma_u) = P(\exists \text{ SATClicks} \in D_{\text{clicked}}^{t-1}) \quad (4)$$

$$P(\omega_t = \text{Non-Rel}|\Sigma_u) = P(\nexists \text{ SATClicks} \in D_{\text{clicked}}^{t-1}) \quad (5)$$

The conditional probability of observations $P(s_t = \text{Rel}|\omega_t = \text{Rel})$ and $P(s_t = \text{Non-Rel}|\omega_t = \text{Non-Rel})$ can be calculated by maximum likelihood estimation. For instance, $P(s_t = \text{Rel}|\omega_t = \text{Rel}) = \frac{\# \text{ of observed true relevant}}{\# \text{ of observed relevant}}$, where “# of observed true relevant” was the number of times where the previously returned document set D_{t-1} contain at least one SAT clicks and those SAT clicked documents were indeed relevant documents in the ground truth. “# of observed relevant” was the number of times where D_{t-1} contains at least one SAT clicks. The ground truth of whether the SG lands on a “Relevant” state was generated by documents whose relevance grades ≥ 3 (relevant to highly relevant). The relevance were judged by assessors from the National Institute of Standards and Technology (NIST) [41].

The second type of observation was related to *exploitation vs. exploration*. This corresponds to a combined observation at Step 3 and the previous Step 6, where the SG update the before-message-belief-state $b_{\bullet\Sigma_{se}}$ for a user action a_u (query

change) and a search engine message $\Sigma_{se}=D_{t-1}$, the top returned documents at the previous iteration. The search engine agent makes observations about exploitation vs. exploration (the second dimension of hidden decision states) by:

$$\begin{aligned} & O(s_t=\text{Exploitation}, a_u=\Delta q_t, \Sigma_{se}=D_{t-1}, \omega_t=\text{Exploitation}) \\ & \propto P(s_t = \text{Exploitation} | \omega_t = \text{Exploitation}) \\ & \times P(\omega_t = \text{Exploitation} | \Delta q_t, D_{t-1}) \end{aligned} \quad (6)$$

$$\begin{aligned} & O(s_t=\text{Exploration}, a_u=\Delta q_t, \Sigma_{se}=D_{t-1}, \omega_t=\text{Exploration}) \\ & \propto P(s_t = \text{Exploration} | \omega_t = \text{Exploration}) \\ & \times P(\omega_t = \text{Exploration} | \Delta q_t, D_{t-1}) \end{aligned} \quad (7)$$

The search engine can guess the hidden states based on the following intuition:

$$s_t \text{ was likely to be } \begin{cases} \text{Exploration} & \text{if } (+\Delta q_t \neq \emptyset \text{ and } +\Delta q_t \notin D_{t-1}) \\ & \text{or } (+\Delta q_t = \emptyset \text{ and } -\Delta q_t \neq \emptyset) \\ \text{Exploitation} & \text{if } (+\Delta q_t \neq \emptyset \text{ and } +\Delta q_t \in D_{t-1}) \\ & \text{or } (+\Delta q_t = \emptyset \text{ and } -\Delta q_t = \emptyset) \end{cases}$$

The idea was that given that D_{t-1} was the message from search engine and $a_u = \Delta q$ was the message from user, if added query terms $+\Delta q$ appear in D_{t-1} , it was likely that the user stays at the same sub information need from iteration $t-1$ to t for ‘exploitation’. On the other hand, if the added terms $+\Delta q$ do not appear in D_{t-1} , it was likely that the user moves to the next sub information need from iteration $t-1$ to t for ‘exploration’. In addition, if there was no added terms ($+\Delta q_t$ was empty) but there were deleted terms ($-\Delta q_t$ was not empty), it was likely that the user goes to a broader topic to explore. If $+\Delta q_t$ and $-\Delta q_t$ were both empty, it means there was no change to the query, it was likely to fall into exploitation.

Hence, $P(\omega_t | \Delta q_t, D_{t-1})$ can be calculated as:

$$P(\omega_t = \text{Exploration} | \Delta q_t, D_{t-1}) = \frac{P(+\Delta q_t \neq \emptyset \wedge +\Delta q_t \notin D_{t-1}) + P(+\Delta q_t = \emptyset \wedge -\Delta q_t \neq \emptyset)}{P(+\Delta q_t \neq \emptyset \wedge +\Delta q_t \in D_{t-1}) + P(+\Delta q_t = \emptyset \wedge -\Delta q_t = \emptyset)} \quad (8)$$

$$P(\omega_t = \text{Exploitation} | \Delta q_t, D_{t-1}) = \frac{P(+\Delta q_t \neq \emptyset \wedge +\Delta q_t \in D_{t-1}) + P(+\Delta q_t = \emptyset \wedge -\Delta q_t = \emptyset)}{P(+\Delta q_t \neq \emptyset \wedge +\Delta q_t \in D_{t-1}) + P(+\Delta q_t = \emptyset \wedge -\Delta q_t = \emptyset)} \quad (9)$$

where D_{t-1} include all clicked documents and all snippets that were ranked higher than the last clicked document at iteration $t-1$. User actions a_u include the current query changes $+\Delta q_t$ and $-\Delta q_t$. In fact, $P(\omega_t | \Delta q_t, D_{t-1})$ needs to be calculated for each specific case. For instance, $P(\omega_t = \text{Exploration} | a = \text{‘delete term’}, \Delta q_t, D_{t-1}) = \frac{\# \text{ of observed true explorations due to deleting terms}}{\# \text{ of observed explorations due to deleting terms}}$. Here we only calculate for the actions with “deleted terms”. “# of observed explorations” was the number of observed explorations suggesting that the user was likely to explore another subtopic based

on Eq. 8, while “# of observed true explorations” was the number of observed explorations judged positive by human accessors in a ground truth. The annotations can be found online.²

The conditional probability $P(s_t = \text{Exploitation} | \omega_t = \text{Exploitation})$ was calculated as $\frac{\# \text{ of observed true exploitations}}{\# \text{ of observed exploitations}}$, where “# of observed exploitations” was the number of observed exploitations suggesting that the user was likely to exploit the same subtopic (based on Eq. 9), and “# of observed true exploitations” was the number of observed exploitations that were judged positive in the ground truth. $P(s_t = \text{Exploration} | \omega_t = \text{Exploration})$ was calculated in a similar way.

At every search iteration the belief state b was updated twice; once at Step 3, another at Step 8. It reflects the interaction and cooperative game between the two agents.

A belief $b_t(s_i)$ was defined as $P(s_i | a_t, b_t)$. The initial belief states can be calculated as: $b_0(s_i = S_z) = P(s_i = S_x)P(s_i = S_y)$, where $x \in \{R = \text{Rel}, NR = \text{Non-Rel}\}$, $y \in \{R = \text{exploRation}, T = \text{exploiTation}\}$, z was the cross-product of x and y and $z \in \{RR, RT, NRR, NRT\}$. In addition, $0 \leq b(s_i) \leq 1$ and $\sum_{s_i} b(s_i) = 1$.

The belief update function was $b_{t+1}(s_j) = P(s_j | \omega_t, a_t, b_t)$ by taking into account new observations ω_t . It was updated from iteration t to iteration $t + 1$:

$$\begin{aligned} b_{t+1}(s_j) &\stackrel{\text{def}}{=} P(s_j | \omega_t, a_t, b_t) \\ &= \frac{O(s_j, a_t, \omega_t) \sum_{s_i \in S} T(s_i, a_t, s_j) b_t(s_i)}{P(\omega_t | a_t, b_t)} \end{aligned} \quad (10)$$

where s_i and s_j were two states, $i, j \in \{RR, RT, NRR, NRT\}$. t indices the search iterations, and $O(s_j, a_t, \omega_t) = P(\omega_t | s_j, a_t)$ was calculated based on what we presented before. $P(\omega_t | a_t, b_t)$ was the normalization factor to keep $\sum_{s_i \in S} b(s_i) = 1$. For notation simplicity, we would only use a to represent actions from now on. However, it was worthy noting that actions can be both domain-level actions a and messages Σ .

Transition probability $T(s_i, a_t, s_j)$ was defined as $P(s_j | s_i, a_t, b_t)$. It was can be calculated as $T(s_i, a_t, s_j) = \frac{\# \text{Transition}(s_i, a_t, s_j)}{\# \text{Transition}(s_i, a_t, s^*)}$, where $\text{Transition}(s_i, a_t, s_j)$ was the sum of all transitions that starts at state s_i , takes action a_t , and lands at state s_j . $\text{Transition}(s_i, a_t, s^*)$ was the sum of all transitions that starts at state s_i and lands at any state by action a_t .

Finally, taking $O(s_j, a_t, \omega_t) = P(\omega_t | s_j, a_t)$, which also equals to $P(\omega_t | s_j, a_t, b_t)$ when we consider beliefs, and $T(s_i, a_t, s_j) = P(s_j | s_i, a_t, b_t)$, the updated belief can

²The manual annotations for “exploration” transitions can be found at www.cs.georgetown.edu/~huiyang/win-win.

be written as:

$$\begin{aligned} b_{t+1}(s_j) &= \frac{P(\omega_t|s_j, a_t, b_t) \sum_{s_i \in S} P(s_j|s_i, a_t, b_t) b_t(s_i)}{P(\omega_t|a_t, b_t)} \\ &= \frac{P(\omega_t|s_j, a_t, b_t) \sum_{s_i \in S} P(s_j|s_i, a_t, b_t) b_t(s_i)}{\sum_{s_k \in S} P(\omega_t|s_k, a_t) \sum_{s_i \in S} P(s_k|s_i, a_t) b_t(s_i)} \end{aligned} \quad (11)$$

where $b_t(s_i)$ was $P(s_i|a_t, b_t)$, whose initial value was $b_0(s_i)$.

After every search iteration, we decide the actions for the search engine agent. we employ Q-learning [38] to find out the optimal action. For all $a \in A_{se}$, we write the search engine's Q-function, which represents the search engine agent's long term reward, as:

$$Q_{se}(b, a) = \rho(b, a) + \gamma \sum_{\omega \in \Omega} P(\omega|b, a_u, \Sigma_{se}) P(\omega|b, \Sigma_u) \max_a Q_{se}(b', a) \quad (12)$$

where the reward for a belief state b was $\rho(b, a) = \sum_{s \in S} b(s) R(s, a)$. $P(\omega|b, a_u, \Sigma_{se})$ corresponds to Eq. 8 and Eq. 9 and $P(\omega|b, \Sigma_u)$ corresponds to Eq. 4 and Eq. 5. b' was the belief state updated by Eq. 11.

In Win-Win search, we take into account both the search engine reward and the user reward. As in [25], we have Q_u calculated as the long term reward for the user agent:

$$\begin{aligned} Q_u(b, a_u) &= R(s, a_u) + \gamma \sum_{a_u} T(s_t|s_{t-1}, D_{t-1}) \max_{s_{t-1}} Q_u(s_{t-1}, a_u) \\ &= P(q_t|d) + \gamma \sum_a P(q_t|q_{t-1}, D_{t-1}, a) \max_{D_{t-1}} P(q_{t-1}|D_{t-1}) \end{aligned} \quad (13)$$

which recursively calculates the reward starting from q_1 and continues with the policy until q_t . $P(q_t|d)$ was the current reward that the user gains through reading the documents. $\max_{D_{t-1}} P(q_{t-1}|D_{t-1})$ was the maximum of the past rewards. The formula matches well with common search scenarios where the user makes decisions about their next actions based on the most relevant document(s) they examined in the previous run of retrieval. Such a document we call it *maximum rewarding document(s)*. we use document with the largest $P(q_{t-1}|d_{t-1})$ as the maximum rewarding document. $P(q_{t-1}|d_{t-1})$ was calculated as $1 - \prod_{t \in q_{t-1}} \{1 - P(t|d_{t-1})\}$, where $P(t|d_{t-1}) = \frac{\#(t, d_{t-1})}{|d_{t-1}|}$, $\#(t, d_{t-1})$ was the number of occurrences of term t in document d_{t-1} , and $|d_{t-1}|$ was the document length.

By optimizing both long term rewards for the user and for the search engine, we learn the best policy π and use it to predict the next action for the search engine. The joint optimization for the dual-agent SG can be represented as:

$$a_{se} = \arg \max_a (Q_{se}(b, a) + Q_u(b, a_u)) \quad (14)$$

where $a_{se} \in A_{se}$ at $t = n$ and n was the number of search iterations in a session, i.e., the session length.

In Win-Win search, A_{se} can include many search engine actions. One type of actions was adjusting a query's term weight. Assuming the query was reformulated from the previous query by adding $+\Delta q$ or deleting $-\Delta q$. That was to say, $A_{se} = \{\text{increasing, decreasing, or keeping term weights}\}$. The term weights were increased or decreased by multiplying a factor. we also use a range of search techniques/algorithms as action options for the search engine agent. Based on Eq. 14, the Win-Win search framework picks the optimal search engine action.

3.1.2 Resolving the Efficiency Issue – Direct Policy Learning

Development in dynamic search algorithms, including ours ([25], [55]) show that using RL for session search was a promising new direction. However, most approaches often trade model complexity off for efficiency. Reducing the number of states and actions was often a must-do to achieve real-time interactions between the user and the search engine. For instance, only four decision-making states were modeled in [55]. In the pursuit of efficiency, an obvious drawback of the prior work was that the details of the rich user-search-engine interactions could get lost.

In this research, we proposed a novel retrieval model for session search. We aim to create a framework that was able to incorporate a variety of features to adequately represent the complex interactions in a session. Our approach was a direct policy learning algorithm that skip the trouble of calculating beliefs for states, but directly learns the optimal policies from observations obtained easily in the user and search engine interactions. During every search iteration, the search engine maximizes the long-term reward towards the ultimate information need, instead of finding a local maximum for the current query. The framework learned a mappings via gradient descent from observations to actions, and sequentially handled a search iteration in three phases, including a browse, a query, and a rank phase. By learning over a history of actions and observations, our algorithm was able to efficiently learn policies with less model complexity than prior work.

Here our states were search iterations. Actions include browsing documents, such as reading and clicking documents, writing queries, and retrieving documents. Every state and action pair receives an immediate reward at the state transitions. We aimed to optimize the long term overall rewards for the entire process.

To solve the RL problem is to find the best policy for it. This was called “policy learning.” The goal of policy learning was to find a policy that optimizes the expected discounted long-term cumulative reward – i.e., the value function

$$V_{\theta}(s_0) = E\left(\sum_{t=0}^{\infty} \gamma^t r(t) | s_0\right) \quad (15)$$

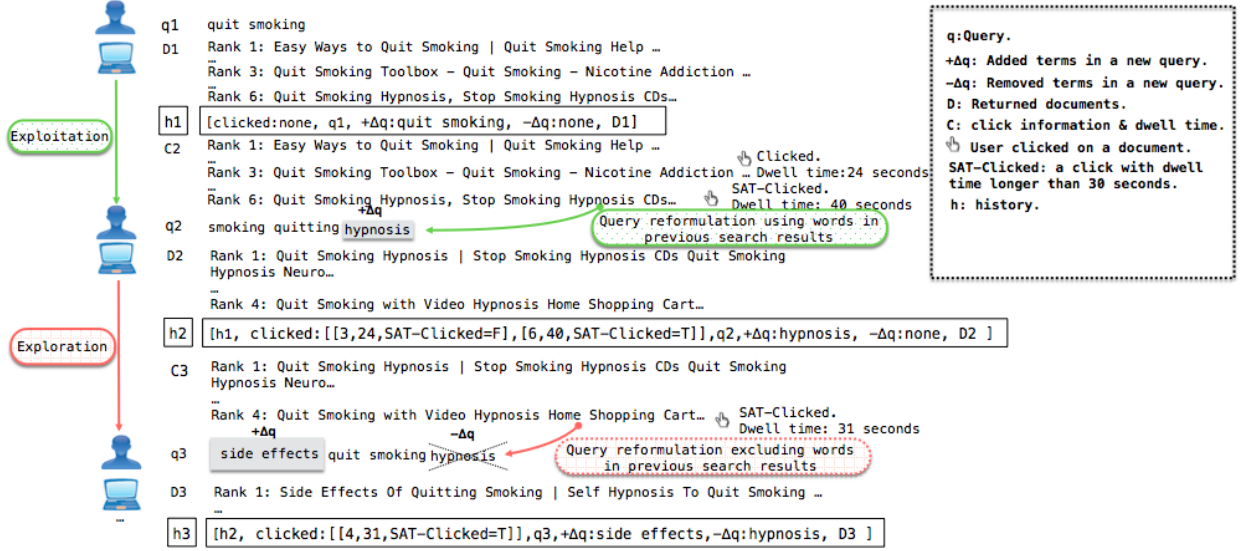


Figure 5: Example History in Session Search (TREC 2014 Session 1011).

where γ was discounts the expected future or past rewards, since they could be inflated now. $r(t)$ was the immediate reward at time t . s_0 was the starting state.

If we denote by H the set of all possible history of search behavior sequences h and $r(t, h)$ the t^{th} reward in the history h , the value function in Eq. 15 can be written as:

$$V_{\theta}(s_0) = \sum_{t=0}^{\infty} \gamma^t \sum_{h \in H} P(h|\theta) r(t, h) \quad (16)$$

where $P(h|\theta)$ was the probability distribution over all possible histories given the parameter vector θ . $r(t, h)$ was the reward obtained at time step t . γ was the discount factor and t indexes the search iterations.

The best policy π^* could be induced from optimization of $V_{\theta}(s_0)$, which was equivalent to learning the best parameter vector θ . The value function can optimized by solving the Bellman equation if the model was known [25]. When the model was not known, there was no close form solution to it and we need to use gradient descent to find the optimal solution.

Two challenges of this method were 1) defining the history and calculating $P(h|\theta)$, and 2) optimizing the value function via gradient descent.

Given that many factors in session search, such as a set of continuous queries, documents, and clicks, it was non-trivial to clearly describe the dynamics in it. To tackle this, we proposed to decompose a search history into two layers. The first

was to break the history into search iterations, indexed by t . Further, an iteration was decomposed into three phases: the browse, the query, and the rank phases. The first two phases were led by the user and the last by the search engine. With such a decomposition, we were able to perform further study to model the search at a more fine-grained level.

We defined a history to be the record of a session from search iteration 0 to the current search iteration t .³ A *history* keeps track of the dynamic changes of states, observations, actions, and rewards during session search. It was a chain of events happening in a session. Every search iteration in the history consists of clicks C , dwell time T , query q , query changes Δq , and retrieved documents D . The history at time t can be written as:

$$h_t = [h_{t-1}, C_t, T_t, q_t, \Delta q_t, D_t]$$

where h_{t-1} was a prefix history in h_t .

Figure 5 illustrates a history gathered for TREC 2014 session 1011. The information need was *You would like to provide your friend with relevant information about: different ways to quit smoking, programs available to help quit smoking, benefits of quitting smoking, second effects*. The session started with the first query (q_1) *quit smoking*. The history up to the end of the first iteration was h_1 , which contains q_1 and a set of documents D_1 returned by the search engine. We assumed a dummy starting query q_0 . Therefore, the new added terms were just q_1 and the removed terms were none.

The second search iteration started when the user browsed the previously returned document set D_1 . The user clicked documents ranked at positions 3 and 6. On the 3rd document d_3 , a dwell time of 24 seconds was spent and on the 6th, 40 seconds. d_6 was considered as a satisfactory click (SAT-Click) since its dwell time was longer than 30 seconds, which was a simple rule that we used to identify SAT-Clicks. The user then entered the next query, q_2 (*quit smoking hypnosis*).

The changes between the new query q_2 and the previous query q_1 , termed as *query changes*, which suggest the users' judgment on document relevance and their degrees of desire to explore. Adopting the definitions in [25], we defined the query changes as the syntactic editing changes between two consecutive queries.⁴ They include positive query change ($+\Delta q$), i.e., the added terms in the new query, and negative query change ($-\Delta q$), i.e., the removed terms from the old query. Here we obtained $+\Delta q_2$ as *hypnosis* and $-\Delta q_2$ as none.

We also noticed that the added query terms (in this case *hypnosis*) can be found in the SAT-Clicked document d_6 ; which suggests that the user could be

³We used 'iteration' and 'time step' interchangeably to refer to search iteration t .

⁴This definition of query reformulation patterns omit fine-grained classifications. However, it works effectively in session search, as reported by [25].

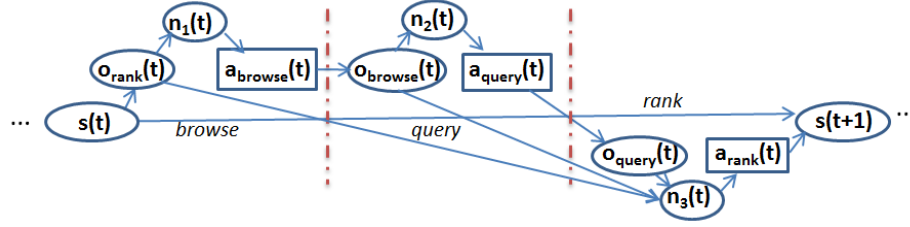


Figure 6: Influence Diagram for Session Search.

inspired by reading the documents retrieved at the previous iteration and issued a new query to find more about it. Since the query reformulation used words in the previously retrieved documents, the transition from the first to the second iteration was identified as an “exploitation” – the user stays at the same subtopic. The search engine then returned documents D_2 for q_2 . The history up to iteration 2 was h_2 , which includes the earlier history h_1 and the new updates in this iteration.

At the third search iteration, the history h_3 also records the earlier histories, clicked documents (the 4th ranked document was SAT-Clicked), the new query q_3 (*side effects quit smoking*), the query changes (added terms $+\Delta q_3 = \text{“side effects”}$ and removed terms $-\Delta q_3 = \text{“hypnosis”}$), and the new documents being returned D_3 . The positive query changes from q_3 to q_2 were not found in the SAT-Clicked document; instead, the term “hypnosis”, which was from the previously returned documents, was removed from q_3 . Because the query reformulation excludes words from the previous search results, we considered this transition as an “exploration” – the user would like to shift to a new subtopic.

Within a search iteration, we could see that the complexity for session search was still high – there involve queries, documents, clicks and query changes, among which many dependencies exist. However, we also noticed that clicks, query, and retrieval in fact happen at different phases. Finer decomposition of the history beyond search iterations was thus possible. Based on such a decomposition, we could obtain a generative model which represents how a session was produced by both the user and the search engine. It would be greatly helpful to explore the interactions between the two agents. In this research, we presented a influence diagram which describes a generative model with further decomposition of the history.

Figure 6 depicts the influence diagram of the observations, actions, and states in session search. The left most and the right most parts of the graph exhibit the states and their transitions, where the states were search iterations and indexed by t . Within an iteration, there were three phases: *browse* phase, *query* phase, and *rank* phase. They were separated by vertical lines in the figure. The upper

part of the figure shows the phases that involves the user, and the lower part the search engine. At each phase, an agent performs the action and after that, an observation was made by the other agent. For instance, a_{rank} at time t (the search engine's ranking algorithm) yields o_{rank} at time $t + 1$ (the document retrieved by the ranking algorithm). The observations would be used at later phases to make decisions for action selection, for instance, o_{rank} influences how a_{browse} should be selected. The observation in the current phase and earlier phases could be stored in an internal states n , which was not a real state that transitions, but a unit to memorize the observations and actions so far. Here n_1, n_2, n_3 were internal states, where 1 indicates browse, 2 query, and 3 rank.

Below were details about the three phases we proposed here.

- *Phase 1: Browse.* The main actor in this phase was the user. The browse phase was after the search results were shown to the user and before the user starts to write the next query. The *browse* phase records how the user perceives and examines the (previously retrieved) search results. From a search engine's perspective, the emphasis was on getting users' feedback on these search results. Most feedback in this phase was implicit, including dwell time T and clicks C . These feedbacks and the derived feedback, such as SAT Clicks [44], were quite reliable relevance judgments from the user and hence can be used to calculate the reward function $r(t, h)$.

Symbols that were used here and related to the browse phase are:

- $o_{rank}(t)$ They were the observations of what the search engine's actions yield. For the previously returned documents at time $t - 1$, we denoted them as D_{t-1} . If the current iteration t was the only search iteration in context, we omitted t from now on, for most symbols.
- a_{browse} were the actions that the user takes to browse D_{t-1} . The actions include clicking and reading documents. The selection of the browsing actions depends on query q_t and retrieved documents D_{t-1} .
- n_1 was an internal state at the user side. It memorizes o_{rank} .
- *Phase 2: Query.* The main actor in this phase was still the user.

The phase happens when the user writes a new query. We assumed that the query was created based on what has been seen in the browse phase, i.e., the previously retrieved documents, and the user's information need about the entire session. The new query triggers another iteration of document retrieval.

Symbols related to the query phase are:

- o_{browse} was the observations about how the user behaves in the browse phase. It includes the click information C_t and dwell time T_t for the previously retrieved documents. Note they were for D_{t-1} the documents retrieved at the previous iteration.
 - a_{query} was q_t , the query that the user writes at the current iteration t .
 - n_2 was another internal state at the user side. It records o_{browse} and helps the user obtain hints from documents being clicked or documents being read for a long time to write the new query q_t .
- *Phase 3: Rank.* The main actor in this phase was the search engine. The rank phase happens after the query was entered and before the search results were returned. The search engine delivers a ranked list of documents based on the current query and other earlier signals being collected. It was the search engine's ranking algorithm and the focus of this research. The retrieved documents would be examined at the beginning – i.e., the browse phase – of the next iteration. The loop continues until the search stopped by the user. In the first iteration, there was no browse phase.

Symbols related to the rank phase are:

- o_{query} was what the search engine observes from what the user wrote as a query. It includes the query q_t and the query changes from q_t to q_{t-1} . The query changes contain two parts, the added terms $+\Delta q_t$ and the removed terms $-\Delta q_t$.
- a_{rank} was the search engine action to retrieve a ranked list of documents. We defined it as the action to put a document d at the top of a ranked list. It depends on a combination of query change Δq_t , previously retrieved documents D_{t-1} , and user clicks C_t and dwell time T_t for D_{t-1} .
- n_3 was an internal state at the search engine side. It memorizes o_{rank} , o_{browse} and o_{query} , to allow the search engine generates optimal actions based on all three types of observations.

The three-phase design of the search history help produce a generative model of how the user and the search engine work together in a session. It has great potentials to study the interactions between the user and the search engine. However, in this research, we focused on creating better search algorithms, therefore only utilizing the search engine part of the generative model.

According to the influence diagram in Figure 6, we wrote the history distribution as

$$P(h|\theta) = \prod_{t=1}^{len(h)} P(o_{rank}(t), a_{browse}(t), o_{browse}(t), a_{query}(t), o_{query}(t), a_{rank}(t) | h_{t-1}, \theta) \quad (17)$$

where $len(h)$ was the length of history h and t indexes the search iterations.

By the chain rule, each factor in the history can be further factorized. In addition, by applying the Markov property and dropping the constants, $P(h|\theta)$ can be reduced into three factors:

$$\begin{aligned} P(h|\theta) &\propto \prod_{t=1}^{len(h)} P(a_{browse}(t) | o_{rank}(t), \theta_1) \\ &\times P(a_{query}(t) | o_{browse}(t), \theta_2) \\ &\times P(a_{rank}(t) | o_{browse}(t), o_{query}(t), o_{rank}(t), \theta_3) \end{aligned} \quad (18)$$

After simplifying Eq. 18 with index i to indicate the phases (1 indicates the browse phase, 2 for query, and 3 for rank):

$$P(h|\theta) \propto \prod_{t=1}^{len(h)} \prod_{i \in \{1,2,3\}} P(a^i(t) | n_i(t), \theta_i) \quad (19)$$

where θ_1 , θ_2 and θ_3 were sub-vectors of θ at each phase. n_1 , n_2 , n_3 were internal states in the POMDP that memorize the observations and actions so far at time t . n_1 stores o_{rank} , n_2 stores o_{browse} , and n_3 stores o_{rank}, o_{browse} , and o_{query} .

The actions of an agent were selected from a range of options. The best actions were expected to be discovered through optimization. To force gradient descent to obey the simplex constraints, we assumed that the actions follow the Softmax function [74]. Specifically, for an action in the i^{th} phase in a search iteration, we defined the action selection distribution as

$$P(a^i | n_i, \theta_i) = \frac{e^{\theta_i \cdot \phi(a^i, n_i)}}{\sum_{a'^i} e^{\theta_i \cdot \phi(a'^i, n_i)}} \quad (20)$$

where ϕ was the feature vector and θ was the parameter vector.

To find the optimal action, that was to obtain the optimal model parameters for θ_1 , θ_2 and θ_3 , we applied gradient descent [10] on the long term reward function,

i.e., the value function. If we took the gradient on the value function, we got:

$$\begin{aligned}
\frac{\partial V_\theta(s_0)}{\partial \theta_k} &= \sum_{t=1}^{\infty} \gamma^t \sum_{h \in H} r(t, h) \frac{\partial P(h|\theta)}{\partial \theta_k} \\
&= \sum_{t=1}^{\infty} \gamma^t \sum_{h \in H} r(t, h) P(h|\theta) \\
&\quad \times \sum_{i=0}^t \frac{\partial \ln[P(a_{browse}|n_1, \theta_1)P(a_{query}|n_2, \theta_2)P(a_{rank}|n_3, \theta_3)]}{\partial \theta_k}
\end{aligned} \tag{21}$$

where H was the set of all the possible histories and for phase i , the gradient was :

$$\frac{\partial \ln P(a^i|n_i, \theta_i)}{\partial \theta_i} = \phi(a^i, n_i) - \sum_{a'} \phi(a'^i, n_i) P(a'^i|n_i, \theta_i) \tag{22}$$

Actions at different phases in a search iteration can all be modeled in a form similar to Eq. 20. In this research, we focused on finding the optimal actions for the search engine. In the rank phase, the search engine agent needs abundant information from the history. The history information, including o_{rank} , o_{query} , and o_{browse} , was stored in n_3 . The ranking actions a_{rank} were selected based on:

$$P(a_{rank}|n_3, \theta_3) = \frac{e^{\theta_3 \cdot \phi(a_{rank}, n_3)}}{\sum_{a'_{rank}} e^{\theta_3 \cdot \phi(a'_{rank}, n_3)}} \tag{23}$$

where θ_3 was a vector of parameters relating to the policy at the rank phase.

Specifically, for the rank phase, the gradient was :

$$\begin{aligned}
\frac{\partial \ln P(a_{rank}|n_3, \theta_3)}{\partial \theta_3} &= \phi(a_{rank}, n_3) - \\
&\quad \sum_{a'_{rank}} [\phi(a'_{rank}, n_3) P(a'_{rank}|n_3, \theta_3)]
\end{aligned} \tag{24}$$

By solving the gradient, we obtained a formula to update θ_3 at each step:

$$\begin{aligned}
\Delta \theta_3 &= \sum_{h \in H} \sum_{t=1}^{\text{len}(h)} \gamma^t r(t, h) \times \sum_{i=1}^t [\phi(a_{rank}, n_3) - \\
&\quad \sum_{a'_{rank}} \phi(a'_{rank}, n_3) P(a'_{rank}|n_3, \theta_3)]
\end{aligned} \tag{25}$$

where H was a set of histories.

Algorithm 1 Direct Policy Learning for Dynamic Search.

```
1: procedure DPL( $H, \epsilon$ )
2:    $\triangleright H$  was the training history set.  $\epsilon$  was a threshold.
3:    $\theta_3 \leftarrow \text{random}(0, 1)$ 
4:
5:   repeat
6:     Sample history  $h$  from  $H$ :
7:      $q_0, D_0, C_0, T_0 \leftarrow \emptyset, r(1, h) \leftarrow 0, \Delta\theta_3 \leftarrow 0$ 
8:
9:     for  $t = 1$  to  $\text{len}(h)$  do
10:       $o_{rank} \leftarrow D_{t-1}, n_1 \leftarrow o_{rank}$ 
11:
12:       $\triangleright a_{browse}$  was performed by the user
13:       $o_{browse} \leftarrow (C_t, T_t), n_2 \leftarrow o_{browse}$ 
14:
15:       $r(t, h) \leftarrow \text{CalculateReward}(D'_t, o_{browse}, h)$ 
16:
17:       $\triangleright a_{query}$  was performed by the user
18:       $o_{query} \leftarrow \text{GetQueryChange}(q_t, q_{t-1})$ 
19:       $n_3 \leftarrow (o_{rank}, o_{browse}, o_{query})$ 
20:
21:      Sample a search engine action  $a_{rank}$ 
22:       $\sim P(a_{rank}|n_3, \theta_3)$ 
23:       $D'_t \leftarrow \text{DocRanking}(a_{rank})$ 
24:
25:       $\Delta\theta_3 \leftarrow \text{UpdateGradient}(r(t, h), D'_t, n_3, \theta_3)$ 
26:       $\theta_3 \leftarrow \theta_3 + \Delta\theta_3$ 
27:    end for
28:  until  $\Delta\theta_3 < \epsilon$  or  $H$  was running out
29: end procedure
```

Next we explain the gradient descent policy learning algorithm, features, document ranking function, and the reward function $r(t, h)$.

Algorithm 1 outlines the direct policy learning algorithm for session search.

The algorithm takes inputs from a set of histories H and a limiting threshold ϵ for gradient descent. The model parameter θ_3 was randomly initialized between 0 and 1.

The algorithm then enters a loop to examine all the histories and to learn the model parameter θ_3 . Inside the loop, a history h was sampled from the set. We

Table 3: Example Features used in Direct Policy Learning.

Notations w was a term t was the current search iteration q_t was the current query q_{t-1} was the previous query $+\Delta q_t$ was the added terms in q_t , comparing to q_{t-1} $-\Delta q_t$ was the removed terms in q_t , comparing to q_{t-1} D_{t-1} was the previously retrieved documents/snippets
Query Features Test if search term $w \in q_t$ and $w \in q_{t-1}$ # of times that a term w occurs in $q_1, q_2, \dots, q_t$ Document Features the length of the content of a document the length of the snippet of a document Query-Document Features Test if a search term $w \in +\Delta q_t$ and $w \in D_{t-1}$ Test if a document d contains a term $w \in -\Delta q_t$ $tfidf$ score of a document d to q_t Click Features Test if there were SAT-Clicks in D_{t-1} # of times a document being clicked in the current session # of seconds a document being viewed and reviewed in the current session Query-Document-Click Features Test if q_i leads to SAT-Clicks in D_i , where $i = 0 \dots t - 1$ Session Features the length of the current session position at the current session

then examined each search iteration in h and obtain values for the observations o_{rank} and o_{browse} , which were about the previously retrieved documents D_{t-1} , and clicks and dwell time for it at time t , C_t and T_t . It also calculates the query changes Δq_t between the queries and stores it in observation o_{query} .

The search engine then calculates the document ranking score, $P(a_{rank}|n_3, \theta_3)$, for all the documents based on the current model parameter θ_3 , and the features. A new ranking of the documents were produced based on $P(a_{rank}|n_3, \theta_3)$, which was calculated by Eq. 23. The new rankings were used to calculate the gradient updates $\Delta \theta_3$.

The loop stops when the gradient update $\Delta \theta_3$ stops changing much or we

examined all the training data.

The features used in our algorithm were $\phi(a_{rank}, n_3)$. They were descriptors of observations o_{browse} , o_{query} , and o_{rank} , to show the interactions among the queries, documents, query changes, and clicks. We used 112 features in this work and show some of them in Table 3.

The features can be classified into a few groups. Query features, document features, and session features were descriptors of the queries, the documents and the session, respectively. Query-document features were statistics about the interaction of the query terms and the previously retrieved documents. Click features were about the clicking information and the dwell time. The most exciting ones were the query-document-click features. They were about the interactions of all three elements. It is worthy noting that our framework was able to take into account as many features as possible.

It is worthy noting that $P(a_{rank}|n_3, \theta_3)$ was the scoring function for a document d . We used $a_{rank} = d$ to denote that the search engine took an action to generate a document list with document d at the top.

$P(a_{rank}|n_3, \theta_3)$ originally presents the probability of selecting a particular action. In the context of session search, we used it to present the probability $P(a_{rank} = d|n_3, \theta_3)$ that selecting d to be put at the top of a ranked list under n_3 and θ_3 at the t^{th} search iteration. A ranked list of document was generated by sorting through the probabilities.

The reward function $r(t, h)$ is to estimate the benefits from an action, specifically a_{rank} . A reward function can guide the search engine throughout the entire dynamic process of session search.

Since session search is a document retrieval task, it's natural that *the reward function was about document relevance*. Our reward function $r(t, h)$ uses the a weighted average of clicks and SAT Clicks as the reward. First, we defined a strong SAT-Click as a click with dwell time longer than 30 seconds, and a weak SAT-Click as a click with dwell time between 10 and 30 seconds. We calculated the reward as

$$r(t, h) = \sum_{d_i \in D_t} \frac{\text{normalized } gain(d_i, t, h)}{\log_2(rank_{d_i} + 1)} \quad (26)$$

where $rank_{d_i}$ was document d_i 's rank position in D_t , and $gain$ was what a document contributes up to time t in history h :

$$gain(d, t, h) = a \times Count(\text{Strong SAT-Click}) + b \times Count(\text{Weak SAT-Click}) \quad (27)$$

where $Count()$ counts the occurrences of the types of clicks in h_t . The weights were normalized by the sum of such clicks in h . Experimentally we set $a = 1$ and $b = 0.5$. This reward function encourages to rank the strong SAT-Clicked documents high in a document list.

Rewards could also be directly generated from user’s relevance assessments when the assessments were available [25, 55]. For instance, the reward function can be nDCG [33] with ground truth. However, in real time interactions, we did not have the ground truth before hand and therefore could not use this reward. In the experiments, we used it as an ideal reward to produce upper bound runs.

In summary, the direct policy learning reduces model complexity and enable us to achieve a more efficient and less complex model. In addition, we were able to flexibly incorporate a wide range of features to describe the rich interactions in session search. Due to the limitation of the setting of the TREC Session Tracks, where a real user and search engine interaction was missing, we could not actually fully apply our modeling to handle both the user side and the search engine side, however, the modeling was general enough to allow future practice of it in experiments with real time interactions. Nonetheless, the framework was shown to be highly effective by evaluations conducted on three years of TREC Session Tracks. It statistically significantly outperformed the TREC best runs in 2012 and 2014.

Due to its trial-and-error nature, session search was an excellent application for reinforcement learning (RL). As part of the efforts to bring RL to IR, this research makes a contribution in studying direct policy learning for session search. We hope this research encourages more future work in dynamic search for high complexity tasks.

3.1.3 Modeling the Two-Way Communication

We realized a collaboration between the user and the search engine in the dynamic search process. In this part of the research, we proposed a novel reinforcement learning style information retrieval framework and develop a new feedback learning algorithm to model user feedback, including clicks and query reformulations, as reinforcement signals and to generate rewards in the RL framework. From a new perspective, we thought session search as a cooperative game played between two agents, the user and the search engine. We studied the communications between the two agents; they always exchange opinions on “whether the current stage of search was relevant” and “whether we should explore now.” Figure 7 shows the messages sending between the two. The algorithm infers user feedback models by an Expectation Maximization (EM) algorithm from the query logs. We compared to several state-of-the-art session search algorithms and evaluated our algorithm on the TREC 2012 to 2014 Session Tracks. The experimental results demonstrates that our approach was highly effective for improving session search accuracy.

From our point of view, we thought both the user and the search engines were autonomous agents who were *equal partners* in their cooperation. Our view was different from prior work on interactive search where the users were the only main

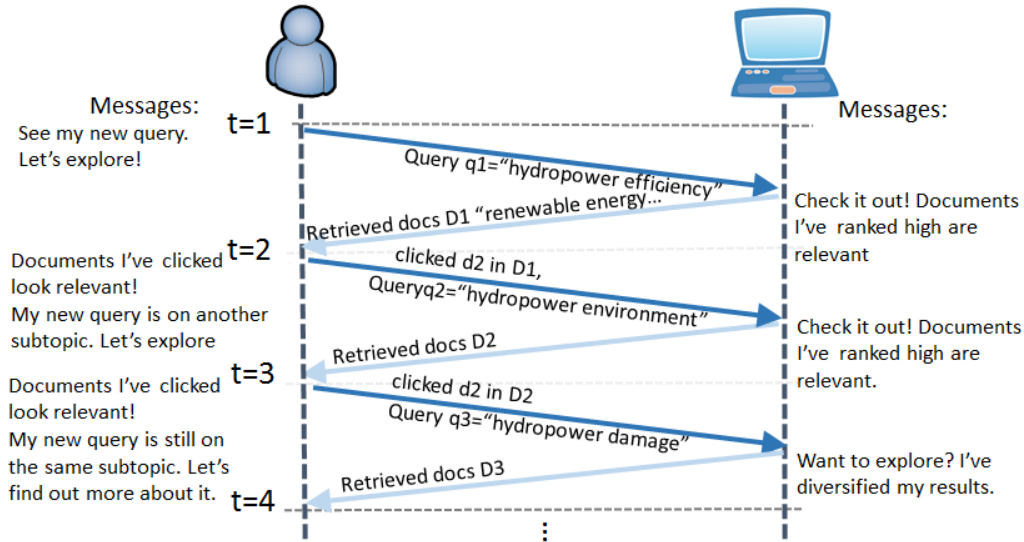


Figure 7: Interactions in Session Search. (Example was from TREC'14 Session 52. d_n was the n^{th} document in a ranked list D at iteration t .)

target in the study. Hence, the roles of the user and the search engine were not that one was teaching and another was learning. Instead, in a joint exploration, neither of them may have a clear picture of how to approach the information need. Sometimes neither of them knows how to describe the information need. For instance, the user may have difficulty articulating a query that precisely describes the information need, especially when the search task was complex. Therefore, we would like to point out that both parties were equal, and that they take turns in exploring the environment and communicating with each other to reach the goal.

Both the user and the search engine could make decisions on relevance, as well as on when and where to explore the information space. A user can issue a new query from one subtopic to another. This drift drives the search process into a new direction. Similarly, the search engine can also drive the search process into new paths, by introducing diversity in the search results that it recommends. In our opinion, both agents were not only able to tell each other what documents they find to be relevant from their own points of views, but were also able to suggest to each other new directions for exploration.

In this research, we proposed a new concept, *universal messages*, which were exchanged in a session between the two agents. Both the user and the search engine communicate and act using these messages. Basically, the two agents use *documents* and other actions, such as queries, to express their opinions on two

Table 4: Message Interpretations.

Messages	“These were Relevant!”	“Let’s Explore!”
User:	“Hey, after reading the documents, those documents that I have clicked or SAT-clicked were potentially relevant. Please pay attention to those documents that I have clicked”	“I have written a query that <i>shifts</i> to new topics. I would like us to explore some new topics together.”
Search Engine:	“Hello, the documents that I’ve ranked high for you were supposed to be relevant. The more relevant I think they are, the higher that I rank them. Please take a look.”	I have a method to recommend novel documents to you as well. When I do it, I introduce some documents on new topics to increase <i>diversity</i> . Please take a look at those documents that I bring to the top positions in the ranking list but may seem not directly related to your current query; they were what I recommend on new topics and hope we explore both of them.”

things: (1) which documents were *relevant*, and (2) whether they would like to inform the other party to *explore* other subtopics as a team. The was to say, there was a two-way feedback mechanism in which the user and the search engine talk to each other every time when they communicate – *whether an agent judges the previously retrieved/recommended documents by the other agent were relevant and whether an agent would like to lead both of them to explore and move to the next sub information need*.

Table 4 shows our interpretations of their messages. We used this intuition to model the two-way feedback between the user and the search engine.

We were fascinated by the complexity of how users generate feedback to the search engine. We also would like to take the challenge to model the search engine as an autonomous agent who behaves as an equal partner to the user; and we modeled its feedback to the user as well. The two-way feedback was modeled as a contextual bandit for the proposed framework.

We have discussed the concept of “universal messaging”. Here we illustrate

more about it. The user’s query can be considered as a piece of short text that attempts to describe the information need. Documents or ranked document list returned by search engine can be considered as a ‘big chunk’ of text that the search engine used to illustrate what in its mind to describe the information need or what were relevant to the information need.

By showing the retrieved documents to the user, which was like showing search engine’s understanding of the information need, the search engine completes its turn of driving the exploration process. The user can also show his or her feedback via clicking documents or spending a certain amount of time to examine the snippets, title or documents. These explicit or implicit feedback from the user can be captured by the search engine. It was the user’s way of expression for what he or she believes that can describes the information need and relevant to the information need.

As shown in Table 4, the messages were about two dimensions: 1) “relevant dimension” – whether an agent thinks the returned documents by the other agent were relevant, and 2) “exploration dimension” – whether an agent would like to drive both agents to explore another subtopic.

To model the various forms of feedback in session search, we defined a decision-making distribution $\mathcal{J}$, which was a distribution over the agents’ decision-making states. We assumed that an agent judges a set of ranked retrieved documents as relevant with probability

$$P(relevant) = 1 - \varepsilon$$

and, the agent judges that it would like to explore with probability

$$P(explore) = \mu$$

At time step t , given the observations o , action a , and target policy π^* , the decision-making distribution $\mathcal{J}$ was defined as:

$$p(\mathcal{J} = RE|o, a, \pi^*) = (1 - \varepsilon)\mu \quad (28)$$

$$p(\mathcal{J} = NRE|o, a, \pi^*) = \varepsilon\mu \quad (29)$$

$$p(\mathcal{J} = RNE|o, a, \pi^*) = (1 - \varepsilon)(1 - \mu) \quad (30)$$

$$p(\mathcal{J} = NRNE|o, a, \pi^*) = \varepsilon(1 - \mu) \quad (31)$$

Here, $\mathcal{J}$ can take four values: ‘RE’ means ‘relevant’ and ‘explore’; ‘NRE’ means ‘non-relevant’ and ‘explore’; ‘RNE’ means ‘relevant’ and ‘non-exploratory’; ‘NRNE’ means ‘non-relevant’ and ‘non-exploratory.’ These correspond to the four decision-making states proposed in [55].

Note that the best policy π^* would take an action a_t , s.t. $\pi^*(o_t) = a_t$. The four formulas sum to one, and the highest value among the four indicates the most likely decision-making state. The corresponding value was used to compose the immediate reward function to guide the reinforcement learning framework.

In our opinion, user feedback was presented in two forms in session search: clicks to indicate relevance, and query change to indicate both relevance and the user's desire to explore.

Here we used time step t to index the search iterations. For query q_t at the t^{th} search iteration, the set of retrieved documents for it was denoted as D_t . The set of documents that were previously retrieved for the previous query at the $(t-1)^{th}$ search iteration was called the previously retrieved documents and denoted as D_{t-1} . It was believed that the previously retrieved documents were very influential in the current run of search, too [25]. They probably influence how the user writes the current query since the user have read some of the documents in D_{t-1} or at least have skimmed the snippets in the list.

The first dimension of judgment was related to *document relevance*.

At search iteration t , if the set of previously returned documents, D_{t-1} , leads to one or more SAT clicks, the current state was likely to be relevant; otherwise it was non-relevant. The intuition was that if while examining D_{t-1} , if the user clicks some documents, it was a good indicator that the user considers the set of the documents that were retrieved by the search engine was relevant at this moment, i.e., at time t . Here we modeled the user clicks as feedback to the previous iteration of search results.

Hence, we modeled the user's underlying judgment on relevance as:

$$\varepsilon = 1 - \frac{\# \text{ of SAT-Clicked documents} \in D_{t-1}}{\# \text{ of returned documents} \in D_{t-1}} \quad (32)$$

where $P(\text{non-relevant}) = \varepsilon$ represents the probability of "non-relevant" cases in the previously retrieved documents D_{t-1} . $P(\text{relevant}) = 1 - \varepsilon$ represents the probability of finding any "relevant" documents in D_{t-1} .

Another dimension of judgment was related to *exploitation vs. exploration*. It was an indicator of how much the user desires to explore.

Here we studied the changes between every consecutive queries as "query change" and denote it as Δq . For the changes between the current query q_t to the previous query q_{t-1} , their query changes at time t was denoted as Δq_t . It was the syntactic difference between the two queries, not the semantic differences. We used it for its simplicity and efficiency for the use in an online setting.

Moreover, the query changes can be grouped into a few categories. They were added terms to the current query, removed query terms from the old query (not in the new query any more), and kept terms in both the current and the old query.

We were especially interested in the added terms and the removed terms, because they somehow suggest the user’s query intent. We denoted them as positive query change $+\Delta q_t$ or negative query change $-\Delta q_t$.

Continue on the idea of the second dimension of judgment. The idea was that given that previously retrieved documents D_{t-1} was the message from the search engine and the current query change Δq_t was the message from the user, if an added query term $+\Delta q_t$ appeared in D_{t-1} , it was very likely that the user stayed with the same subtopic from iteration $t - 1$ to t . It was a state that we called ‘exploitation.’

On the other hand, if the added term $+\Delta q_t$ did not appear in D_{t-1} , it was quite likely that the user has moved to the next subtopic from iteration $t - 1$ to t . It was a state that we called ‘exploration.’

We modeled the user’s underlying judgment on “exploitation vs. exploration” as:

$$\mu = 1 - \frac{\# \text{ query changes } \in D_{t-1}}{\# \text{ of permutations of query terms } \in q_t} \quad (33)$$

where $P(\text{exploration}) = \mu$ represents the probability of “exploratory” cases at time t , and $P(\text{exploitation}) = 1 - \mu$ represents the probability of “exploitation” at time t .

In addition, there were 10 documents displayed to the user in the first 10 search results (SERP). Only one out of the 10 receives a SAT-click.

Therefore, for this iteration, we had $\epsilon = 1 - \frac{1}{10} = 0.9$ and $\mu = 1 - \frac{1}{3!} = 0.83$.

Likewise, the search engine’s feedback was also presented in two forms: One to indicate what the search engine thinks were *relevant*, and another to indicate what the search engine desires to explore in the process.

For the first dimension, we proposed that the search engine uses the top-scored or top-ranked documents ranked by itself, to indicate relevance.

It means that we could use the similarity scores that the search engine assigns to each retrieval result to indicate the search engine’s judgment on relevance. The assumption was that across different queries, query length normalization was taken care of, and the search engine was able to measure query-to-document similarity in a relatively consistent manner. However, in the datasets that we experimented on, such similarity scores were not available. Therefore, instead of using the actual ranking scores, we used the Maximum Likelihood Estimation (MLE) of relevant documents among the top ranked documents to present search engine’s judgment on this dimension.

At the t^{th} search iteration, we modeled the search engine’s underlying judgment on relevance by leveraging the probability that a top returned document was indeed relevant, which means the document appears in the ground truth was a relevant document.

The estimation of ε for search engine was

$$\varepsilon = 1 - \frac{\# \text{ of relevant documents in the top } n \text{ retrieved}}{n} \quad (34)$$

Here, $P(NR) = \varepsilon$ represents the overall probability of “non-relevant” cases at time t . $P(R) = 1 - \varepsilon$ represents the probability of finding any “relevant” documents at time t in the top n returned documents. n was empirically set to 5 in our experiments.

The second dimension was about *exploration*. Here we considered that the search engine introduces diversity in the retrieval results, to indicate the search engine’s desire to explore.

To express its feedback/opinion on whether to *exploit or explore*, the search engine shows different degrees of diversity in its returned results. The idea was that the search engine can return diversified results – i.e. multiple topical clusters in its results – to promote “exploration.” We could detect this by observing the word distribution and indirectly measure the diversity in the top returned snippets. According to the Zipf’s law, assuming that the top returned documents or snippets were on a single topic, it was expected that the slope of the word distribution would be steeper. On the other hand, if the documents were diversified, it was expected that the slope of the word distribution would be flatter.

We therefore modeled the search engine’s underlying message on exploration as

$$\mu = 1 - \frac{\text{total } \# \text{ of the top } m \text{ frequent non-stop-words in } D_t}{\text{total } \# \text{ of non-stop-words in } D_t} \quad (35)$$

Here, $P(\text{exploration}) = \mu$ represents the overall probability of how “exploratory” the results were by measuring the percentage of top m occurring words in all words returned in the search results. Here we only examined the first 10 snippets since they were the content that the search engine would like to use to influence the user at the first sight. They were what the search engine presents in front of the user, and would like to use them to influence the user the most to do a cooperative exploration together. We empirically set m to 5 words.

Here we show an example to show the details. In the second iteration of Session 1011, 8 out of the top 10 documents retrieved by the search engine were highly relevant, as provided in the ground truth. Therefore, $\epsilon = 1 - \frac{4}{5} = 0.2$.

Moreover, there were 428 non-stop-words totally in the top 10 snippets, and the most frequent 5 words are:

“smoke”(59), “quit”(34), “hypnosis”(30), “stop”(19), “button”(7)

The total counts of these 5 most frequent non-stop-words {“smoke”, “quit”, “hypnosis”, “stop”, “button”} was 123. Therefore, $\mu = \frac{123}{428} = 0.29$.

Algorithm 2 Feedback Learning by EM.

```
1: procedure FEEDBACKLEARNING( $\pi, h$ )
2:    $\pi \leftarrow \text{randomPolicy}()$ 
3:    $\varepsilon \leftarrow \text{random}(0, 1), \mu \leftarrow \text{random}(0, 1)$ 
4:
5:   while the search session has not finished do
6:
7:      $o_t \leftarrow \text{observe}()$ 
8:      $a_t \leftarrow \pi(o_t)$ 
9:      $h_t \leftarrow [h_{t-1}, (o_t, a_t)]$ 
10:
11:    repeat
12:      E-Step:
13:      compute  $P(\mathcal{J}|o_t, a_t, \pi)$  by Eq. 28 to Eq. 31
14:       $j = \arg \max_{j' \in [\cdot, RE', \cdot, NRE', \cdot, RNE', \cdot, NRNE'] } P(J = j'|o_t, a_t, \pi)$ 
15:       $r(t, h) = P(\mathcal{J} = j|o_t, a_t, \pi)$ 
16:       $\pi' \leftarrow \arg \max_{\pi} \int_0^1 \int_0^1 p(\varepsilon, \mu|h, \pi) \ln p(h, \varepsilon, \mu|\pi) d\varepsilon d\mu$ 
17:       $a'_t = \pi'(o_t)$ 
18:      performAction(agent,  $a'_t$ )
19:      M-Step:
20:      compute  $\varepsilon'$  and  $\mu'$ 
21:      until  $\varepsilon = \varepsilon'$  and  $\mu = \mu'$ 
22:       $t \leftarrow t + 1$ 
23:    end while
24:  end while
25:   $\pi^* = \pi$ 
26:  output  $\pi^*$ 
27: end procedure
```

Based on the estimation for the two dimensions, the maximum judgment distribution value was 0.568 which means "relevant and exploitation" for this iteration, which was correct based on our manual evaluation. We proposed to use Expectation Maximization [20] to compute the maximum likelihood estimation (MLE) of the decision-making distribution.

Algorithm 2 outlines the EM algorithm. The EM algorithm starts with a random/initial estimation of the two hidden parameters, ε and μ , and an initial random policy, which generates random actions for both agents given an observa-

tion. For the history up to the t^{th} search iteration, we were able to go into the history and go through iteration by iteration to update the policy.

In the Expectation step, we computed the decision-making distribution based on Equations 28 to 31. The most likely joint decision by both dimensions were indexed by j . The reward function $r(t, h)$ was calculated by the decision-making distribution. A new policy was estimated by finding the best policy at step t given the current estimates of ϵ and μ , at line 13. At line 14, an action was selected based on the new estimate of the policy, and at line 15, the corresponding agents, either the user or the search engine, performed the action to the documents. If it was the search engine's turn, it retrieved documents. If it was a user's turn, the user clicked or browsed the documents.

In the Maximization step, we re-computed ϵ and μ at time t based on the new estimate of π' and the new actions yielded from π' , by using the formulas 32 to 35.

The loop repeats until the history was running out or reaching convergence. The algorithm then outputs the final learned policy π^* .

Our solution to the feedback-learning framework adopted multiple-agent reinforcement learning. EM algorithms can often be guaranteed to converge to local minimum points. When there were two (or more) agents in the process, in theory, they were able to jointly reach the local optima for a common solution, which was known as the Nash equilibrium in game theory.

To reach Nash equilibrium, it was assumed that all agents were rational and try their best to reach own local optima. This assumes that the user was rational and always produces good queries to show the search intent. However, the assumption was often not true. We saw many non-optimal user actions, which impact negatively on the retrieval.

For instance, TREC 2014 Sessions 232 and 89 were two session sharing the same information need – “*You were writing a summary article about the Pocono Mountains region. Find as many relevant articles as you can describing the region, things to see and do there ...*” In Session 232, the user submitted three queries: “pocono mountains,” “mountains,” and “mountains.” We could see that the second and the last query were not very informative. It turns out that the number of relevant documents retrieved in the top 10 results drops from 6 to 4 when transitioning from the first query to the rest. It shows that when a user did not work hard enough to produce reasonable and cooperative queries, it was not surprising to see a negative impact on retrieval results. This suggests that it was important for both parties to commit to cooperative activities in session search. In the future, our research could potentially be used to detect user behaviors that could be explained by rational models and develop new user models from there.

On the contrary, in Session 89, the user submitted “Pocono Mountains”, “Pocono Mountain parks”, “Pocono Mountain things to do”, “Pocono Mountain

community”, “Pocono Mountain community”. Among the 6 relevant search results from the first query, the user clicked and examined “clueweb12-0304wb-75-03048” for 23.77 seconds. The document talks about “... *State National Parks Forests ... The Camelbeach Mountain Waterpark has hours of family-friendly activities for you. ... Hickory Run State Park and Boulder Field has acres of land sure to break in the toughest of hiking footwear ...*” The word “park” appears multiple times and probably inspired the user to create the second query “Pocono Mountain parks.” The matching between the query change and the previously retrieved documents was interpreted as “exploitation” in our model, which generates positive reward for $r(t, h)$ as if the user performs optimally. As a result, in this run of search, the number of relevant documents in top 10 boosts from 6 to 8. It suggests that when the user behaviors align well with our models, i.e., the user acts optimally, the cooperation between the user and the search engine functions well and yields good retrieval results. It leaves opportunities for future work in query suggestion.

In summary, we studied the interactions in session search from a new perspective. We modeled the user and the search engine as two equal partners who collaborate and explore together to achieve a common goal – satisfying the information need. Based on this assumption, we were the first to recognize the two-way feedback messages from both the search engine and the user, such as documents retrieved, user clicks and query formulations, as rewarding signals to guide session search in a reinforcement learning framework.

By modeling the two-way feedback from both the relevance and the exploration perspectives, we successfully integrated the rich forms of feedbacks in sessions, both explicit and implicit, and use them as rewards to enhance search accuracy in session search. The proposed EM algorithm provides an initial attempt to formalize the research along this direction. In the experiment, we compared our approaches’ immediate search accuracy with the state-of-the-art session search algorithms over three years of TREC Session Track data. The results show that our approach greatly improves the immediate search accuracy by effectively utilizing the abundant feedback.

3.1.4 Bandits Approaches

Another approach that we explored during the Memex program for dynamic search was the contextual bandits approach. They were stateless and lighter weight than POMDPs. In this research, we proposed using the context surrounding the current search iteration to aid the system through the use of a contextual bandit algorithm. We ran our system on the TREC-DD 2016 Ebola dataset, and compared our system’s runs with baselines extracted from the TREC-DD 2016 submitted runs. We also analyzed the search policy created by our system to better understand

how the system navigated through the dynamic search. Our proposed system shifts more of the search responsibility from the user to the system, leading to a smarter and more agile search system.

Contextual bandit algorithms [88] were a type of reinforcement learning algorithm that uses the context given at each iteration to select an action that would maximize some sort of reward. Contextual bandits are similar to the multi-armed bandit with the augmentation of a historical context that was utilized alongside the reward from previous iterations to select an action. The context can be historical data that was collected since the start of the process, such as in Shivaswamy and Joachims [68], or use the contextual features of the current action to influence the bandit selection policy, such as in LinUCB by Lihong et al. [52].

The LinUCB algorithm was originally designed for personal news recommendation. LinUCB uses contextual information gathered from the user and news articles to estimate a reward for each action. At each iteration, they pick the action that maximizes their estimated reward, using

$$a_t \stackrel{\text{def}}{=} \underset{a \in A_t}{\operatorname{argmax}} \left(\mathbf{x}_{t,a}^\top \hat{\theta}_a + \alpha \sqrt{\mathbf{x}_{t,a}^\top \mathbf{A}_a^{-1} \mathbf{x}_{t,a}} \right) \quad (36)$$

where a_t signifies the selected action, $\mathbf{x}_{t,a}$ was the feature vector with an unknown but estimated coefficient vector $\hat{\theta}_a$ and α was a constant. $\sqrt{\mathbf{x}_{t,a}^\top \mathbf{A}_a^{-1} \mathbf{x}_{t,a}}$ was the standard deviation of the expected reward, where $\mathbf{A}_a^{-1}$ was the inverse of a matrix made by adding a design matrix corresponding to the previous contexts to an identity matrix the size of the context. In [52], they use click data as reward.

We adapted the LinUCB algorithm as its use of context at each iteration to select an action fits well with the multi-iterative approach of a dynamic search process. Our adaptation of LinUCB emphasizes the use of the context surrounding the search iteration, such as what the user found relevant and where previous actions have explored in the information space, to select what the appropriate next action should be. Furthermore, while [52] defines its action as the actual news article that was selected, we defined actions as the tactics used to edit the query to retrieve documents, with the feedback information returned from the simulated user as the reward.

Following actions were used in this contextual bandits model. The action “add” provides more detail to the search by adding another word to the query in order to explore a new, more specific subtopic in the search. We used a form of tf-idf to select the word:

$$s_w = \underset{w \in D_{rel}}{\operatorname{argmax}} \left(\frac{freq_{w,d}}{len(d)} * \ln \left(\frac{size(D_{rel})}{docfreq_w} \right) \right) \quad (37)$$

The score s_w of any given word was the maximum tf-idf score received for that word w in any document d in the set of relevant documents D_{rel} retrieved that iteration. In equation 37, the term frequency was calculated using $freq_{w,d}$, the frequency of a word in a document, divided by the document length, and inverse document frequency was calculated using the natural log of the total number of relevant documents that iteration divided by $docfreq_w$, the number of documents containing that word. The word with the highest score was considered to be the most representative word found in the documents with positive feedback, and was therefore selected to be added to the query.

The action “remove” also applies tf-idf like “add” to select which word to remove. This action widens the search from the previous subtopic. Notably, while the “add” action uses the set of relevant documents, “remove” uses the set of irrelevant documents in the feedback, or D_{nonrel} instead of D_{rel} . This generates a set of least relevant words retrieved in the last iteration. We then selected the word in the current query that has the highest score in the set to be removed.

The action “remove” demonstrates this as the removal of the term “existing,” which widens the search from searching about the currently existing robots being used to aid health care workers to all robots both existing and merely proposed to aid in the Ebola outbreak.

The action “weight” modifies the weights to each query term. This can be considered a continued search on the current subtopic with a shift in its focus. We calculated the adjustment as $weight_{t-1} \pm \alpha(rel(w))$, such that it was addition if the term was relevant and subtraction if irrelevant, and where $\alpha < 1$ was a constant and $rel(w)$ was the degree to which the term w was relevant or irrelevant. We decided the degree of a term’s relevance or irrelevance by creating two lists—the top 20 most frequent words in the relevant documents and that in the irrelevant documents. A term’s relevance or irrelevance was proportional to its rank on either list. Using mathematical representation, a term’s weight was adjusted according to eq. 38, with α set as 0.2, meaning a maximum of 0.2 was added (if relevant) or subtracted (if irrelevant) from the weight at any one time.

$$w_t = \begin{cases} w_{t-1} + .2(\frac{20-rel(w)}{20}) & \text{if word } w \text{ was relevant} \\ w_{t-1} - .2(\frac{20-rel(w)}{20}) & \text{if word } w \text{ was irrelevant} \end{cases} \quad (38)$$

While most dynamic search systems being proposed set a stopping condition for the search manually, either through the number of iterations or number of relevant documents seen, we proposed giving the bandit an action “stop” to decide the search’s stopping condition automatically.

The action “stop” ends the search of one topic and, if applicable, starts the next one. For example, once the bandit has searched the topic “alleged alternative Ebola treatments” adequately and found enough information, the bandit

Table 5: Featured used for Contextual Bandits

#	context feature	explanation
1	# of documents seen	how far search was
2	# of relevant documents last iteration	current effectiveness
3	# of relevant documents seen	depth of search
4	# of times action weight called	search path direction
5	# of times action add called	search path direction
6	# of times action remove called	search path direction

would select the action “stop” and end the search on that topic. The bandit would then start the search on the next topic in the list provided by TREC, “urbanization/urbanisation.”

A reward was given to denote the effectiveness of an action. In a search process, the reward for an iteration can be considered the amount of information gained at that iteration. Therefore, we defined the reward r_t for a selected action at iteration t as the sum of the relevance score RS for each relevant passage p in the set of all documents D retrieved at that iteration.

$$r_t = \sum_{p \in D} RS_p \quad (39)$$

The relevance score for a relevant passage was a score from 1 to 5, with 5 meaning extremely relevant and 1 being only marginally relevant. Note that the reward was never negative but it can be 0 if there were no relevant passages, which would show that the action was not effective in retrieving relevant documents for that iteration. The reward for “stop” was calculated using the set of documents retrieved by the next topic’s initial search.

The context given to the LinUCB model consists of six features, listed in table 5. These features span different aspects of the previous iteration(s) in order to provide the bandit with more information about the current state of the search and search path. Using the reward and the context, the bandit selects the action to play at each round that would maximize the reward.

In summary, we proposed alternative solution to the dynamic search task that uses the context of the current search state in order to determine the next action in the search. Based on our system’s interaction with the TREC-DD Ebola domain, we found that taking into account the context of the current search was relevant towards improving the search’s effectiveness over time. In the future, we can continue to improve these results by applying more advanced query expansion and

Table 6: TREC 2011-2014 Session Track Dataset Statistics.

	2011	2012	2013	2014
#Sessions	76	98	87	1021
#Search topics	62	48	49	60
#Queries	280	297	442	4226
Avg. session length	3.68	3.03	5.08	4.14
Max session length	11	11	21	16
Avg. #sessions per topic	1.23	2.04	1.78	17.02
#Sessions w/ length <4	47	75	38	466
#Sessions w/ length 4~10	28	22	42	543
#Sessions w/ length >10	1	1	7	12

weight adjustment techniques to increase the bandit’s search effectiveness over time. Further research can also be done on the search completion and next topic selection portion of the search, in order to build an even more advanced search system.

3.1.5 Results and Discussion

We evaluated the proposed framework on TREC 2011 – 2014 Session Tracks [39, 40, 41, 42].

Here $DCG@k = \sum_{i=1}^k \frac{2^{rel_i}-1}{\log(1+i)}$, where i was the document’s ranking in the retrieval list, rel_i was the i^{th} document’s relevance grade, and k was the cut off value meaning that the metric assumes that users examine the ranked list from top to bottom and stop at the k^{th} ranking. Then $nDCG@k = \frac{DCG@k}{IDCG@k}$ was just the normalized version of $DCG@k$, where $IDCG@k$ was the ideal $DCG@k$ score calculated from the ideal ranked list, where the most relevant documents were retrieved at the highest rankings. $nDCG$ without parameter k means that the $nDCG$ score was accumulated until the last document in the ranked list, which was the 2000th document in this experience setting.

$nERR@k$ was similar to $nDCG@k$. Let $P(i)$ denotes the relevance probability of a document at rank i . Here $P(i) = \frac{2^{rel_i}-1}{2^{max_grade}}$, and max_grade was the maximum possible relevance grade in the ground truth. The probability that a user doesn’t find the relevant document from document rank 1 to rank p was $P_{dis_satisfy}(p) = \prod_{j=1}^p (1 - P(j))$. Then $ERR@k$ was defined based on the expected probability that the user find the relevant document at rank k . $ERR@k = \sum_{i=1}^k \frac{P(i)P_{dis_satisfy}(i-1)}{i}$. And $nERR@k$ was the normalized version of $ERR@k$ based on the same ideal ranked list as in $nDCG@k$.

AP was the average precision of all relevant documents. It was defined as

Table 7: Search accuracy on TREC 2011 Session ($\dagger$ indicates a statistical significant improvement over Rocchio-CLK at $p < 0.05$ (t-test, one-sided)); * indicates a statistical significant improvement over QCM+DUP at $p < 0.05$ (t-test, one-sided)).

Approach	nDCG@10	nDCG	MAP	nERR@10
Lemur	0.3642	0.2873	0.1202	0.4555
TREC median	0.3407	0.2577	0.1198	0.4155
TREC best	0.4540	0.4118	0.1909	0.5519
PRF	0.3547	0.2753	0.1184	0.4158
Rocchio	0.3681	0.2900	0.1270	0.4338
Rocchio-CLK	0.4126	0.3221	0.1559	0.4980
Rocchio-SAT	0.3921	0.3052	0.1395	0.4781
QCM+DUP	0.4161	0.3118	0.1451	0.4865
QCM SAT	0.4230	0.3175	0.1531	0.4928
Win-Win	0.5252* $\dagger$	0.3651* $\dagger$	0.1715* $\dagger$	0.6272* $\dagger$

Table 8: Search accuracy on TREC 2012 Session ($\dagger$ indicates a statistical significant improvement over Rocchio-CLK at $p < 0.05$ (t-test, one-sided)); * indicates a statistical significant improvement over QCM+DUP at $p < 0.05$ (t-test, one-sided)).

Approach	nDCG@10	nDCG	MAP	nERR@10
Lemur	0.2622	0.2745	0.1342	0.3004
TREC median	0.2344	0.2241	0.1214	0.2831
TREC best	0.3153	0.2613	0.1557	0.3576
PRF	0.2614	0.2634	0.1293	0.2971
Rocchio	0.2489	0.2561	0.1226	0.2924
Rocchio-CLK	0.3439	0.3120	0.1610	0.3832
Rocchio-SAT	0.2864	0.2836	0.1440	0.3265
QCM+DUP	0.3368	0.3052	0.1536	0.3880
QCM SAT	0.3351	0.3047	0.1533	0.3867
Win-Win	0.3631* $\dagger$	0.3316* $\dagger$	0.1651*	0.4222* $\dagger$

Table 9: Search accuracy on TREC 2013 Session ($\dagger$ indicates a statistical significant improvement over Rocchio-CLK at $p < 0.01$ (t-test, one-sided)); * indicates a statistical significant improvement over QCM+DUP at $p < 0.01$ (t-test, one-sided)).

Approach	nDCG@10	nDCG	MAP	nERR@10
Lemur	0.1043	0.1102	0.0398	0.1409
TREC median	0.1531	0.1357	0.0574	0.2049
TREC best	0.1706	0.2445	0.1008	0.2259
PRF	0.0764	0.0971	0.0339	0.1045
Rocchio	0.0783	0.1000	0.0348	0.1130
Rocchio-CLK	0.1161	0.1253	0.0430	0.1697
Rocchio-SAT	0.1002	0.1100	0.0409	0.1311
QCM+DUP	0.1555 $\dagger$	0.1380 $\dagger$	0.0568 $\dagger$	0.2229 $\dagger$
QCM SAT	0.1440 $\dagger$	0.1212	0.0494	0.2111 $\dagger$
Win-Win	0.2137* $\dagger$	0.1639* $\dagger$	0.0632* $\dagger$	0.3147* $\dagger$

Table 10: Search accuracy on TREC 2014 Session ($\dagger$ indicates a statistical significant improvement over Rocchio-CLK at $p < 0.01$ (t-test, one-sided)); * indicates a statistical significant improvement over QCM+DUP at $p < 0.01$ (t-test, one-sided)).

Approach	nDCG@10	nDCG	MAP	nERR@10
Lemur	0.2169	0.1837	0.0227	0.2730
TREC median	0.1881	0.1620	0.0261	0.2183
TREC best	0.2482	0.2218	0.0366	0.3187
PRF	0.2310	0.1890	0.0249	0.2880
Rocchio	0.2351	0.1925	0.0248	0.2968
Rocchio-CLK	0.2407	0.1960	0.0248	0.3128
Rocchio-SAT	0.2051	0.1749	0.0230	0.2546
QCM+DUP	0.2422	0.1881	0.0254	0.3112
QCM SAT	0.2419	0.1884	0.0254	0.3116
Win-Win	0.3076* $\dagger$	0.2339* $\dagger$	0.0282	0.3916* $\dagger$

$AP = \frac{\sum_i (Precision(i) * rel(i))}{\text{total \# of relevant documents}}$, where $rel(i) = 1$ if and only if the document at ranking i was relevant. It was an indicator function. MAP was just the mean AP over all testing sessions. Note that the $nDCG$ and $nERR$ reported in result tables were also mean values over all testing sessions.

$nDCG@k$, $nERR@k$ and MAP were all ranged from 0 to 1. A ranked list with relevant documents at higher rankings would have larger $nDCG@k$, $nERR@k$ and MAP scores. Comparing to $nDCG$ and MAP , $nERR@k$ was highly sensitive to the ranking of the first retrieved relevant document. Also $nDCG@k$ and $nERR@k$ consider document's relevance grade, however MAP treats all relevant documents equally important.

The ground truth relevant documents were provided by TREC. Note that in TREC 2011, the ground truth data provides subtopic level relevance judgment. We used a document's maximum subtopic relevance grade as its relevance grade when we calculated $nDCG@10$ and other metric scores.

We compared our system with the following systems: Lemur [50] (language modeling + Dirichlet smoothing), PRF (Pseudo Relevance Feedback in Lemur assuming the top 20 documents were relevant [12]), Rocchio (relevance feedback that assumes the top 10 previous retrieved documents were relevant [37]), Rocchio-CLK (implicit relevance feedback that assumes only previous clicked documents were relevant), Rocchio-SAT (implicit relevance feedback that assumes only previous SAT-clicked documents were relevant), QCM+DUP (the QCM approach proposed by [25]), and QCM SAT (a variation of QCM by [86]). QCM+DUP increases theme terms' weights, decreases added terms' weights and decreases removed terms' weights. The parameters α , β , δ , γ were set as 2.2, 1.8, 0.4, and 0.92 respectively according to the experiments in [25].

We chose Rocchio-CLK (a state-of-the-art interactive search algorithm) and QCM+DUP (a state-of-the-art dynamic search algorithm) as two baseline systems and all other systems were compared against them. TREC median and TREC best scores were also included for reference. Note that TREC best was an aggregation from the best scores of each individual submitted TREC runs; it was not a single search system. This also applies to TREC median.

The "Win-Win" run implements six retrieval technologies. They are: (1) increasing weights of the added terms ($+\Delta q$) by a factor of $x = \{1.05, 1.10, 1.15, 1.20, 1.25, 1.5, 1.75 \text{ or } 2\}$; (2) decreasing weights of the added terms by a factor of $y = \{0.5, 0.57, 0.67, 0.8, 0.83, 0.87, 0.9 \text{ or } 0.95\}$; (3) the term weighting scheme proposed in QCM+DUP [25]; (4) a PRF (Pseudo Relevance Feedback) algorithm which assumes the top p retrieved documents were relevant while p ranges from 1 to 20; (5) an *ad-hoc* variation of Win-Win, which directly uses the last query in a session to perform retrieval; and (6) a *brute-force* variation of Win-Win, which combines all queries in a session, extracts all unique query terms from them, and weights them

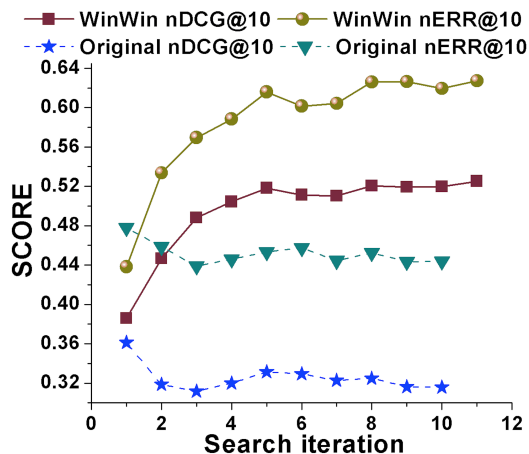


Figure 8: TREC 2011 Immediate Search Accuracy.

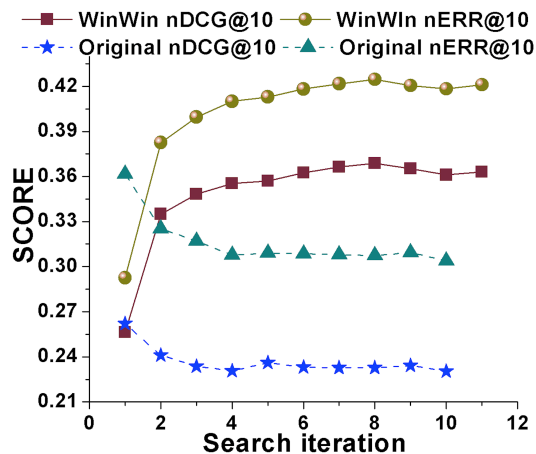


Figure 9: TREC 2012 Immediate Search Accuracy.

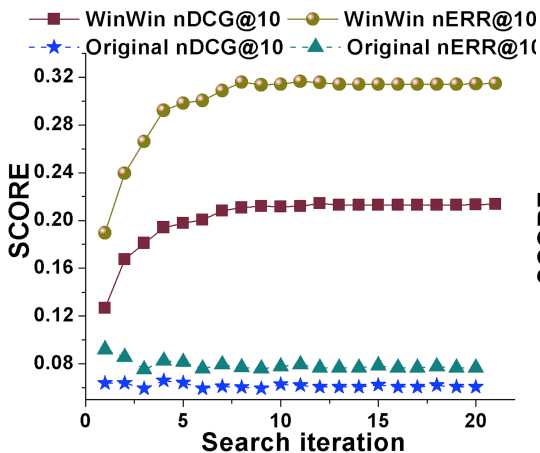


Figure 10: TREC 2013 Immediate Search Accuracy.

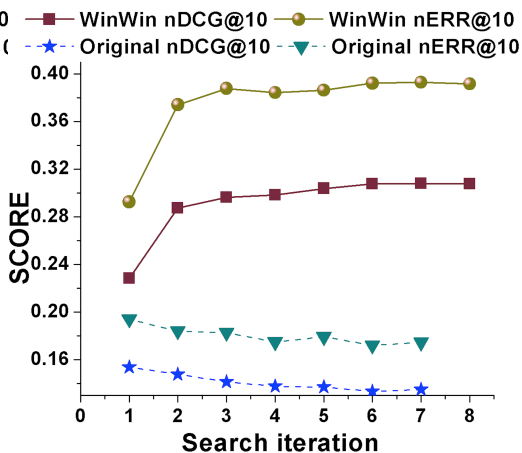


Figure 11: TREC 2014 Immediate Search Accuracy.

equally. Win-Win examine 21 search engine action options in total to find out the optimal action that maximizes the joint long term reward $Q_{se}(b, a) + Q_u(b, a_u)$ for both agents.

Table 7 shows the search accuracy of all systems for TREC 2011 Session Track. We noticed that Win-Win search outperforms all other systems. It statistically significantly outperforms Rocchio-CLK by 27.3% and QCM+DUP by 26.2% in terms of nDCG@10 scores. The trends for other evaluation metrics were similar to nDCG@10.

Table 8 shows the search accuracy of all systems under comparison for TREC 2012 Session Track. Win-Win search was the best system among all systems. It statistically significantly outperforms Rocchio-CLK by 5.6% and QCM+DUP by 7.8% and Lemur by 38.5% in nDCG@10. The trends for other evaluation metrics were also similar to nDCG@10.

Table 9 shows the search accuracy of all systems for TREC 2013 Session Track. Since we only indexed ClueWeb12 CatB, after spam reduction, many relevant CatA documents were not included in the CatB collection. Hence the accuracy performance of all systems were lower in TREC 2013 dataset than in other TREC datasets. Nonetheless we found that Win-Win was still the best run among all systems. It shows statistically significant gain (p-value: .01, one-sided t-test) over all other systems across all evaluation metrics. Particularly, the proposed approach achieves a significant 37.4% improvement of nDCG@10 comparing to QCM+DUP.

Finally, in TREC 2014 Session Track dataset (Table 10), Win-Win also outperforms other baselines statistical significantly. We noticed the trend was consistent across TREC 2011 – 2014 Session Track datasets. The experimental results support that our approach was highly effective.

TREC Session tasks request for retrieval results for the last query in a session. Theoretically, however, Win-Win search can optimize at every search iteration throughout a session. We hence compared our approach (the Win-Win run) with the top returned documents provided by TREC (the Original run) in terms of *immediate search accuracy*. We defined immediate search accuracy at i as an evaluation score that measures search accuracy at search iteration i . The evaluation scores used were nDCG@10 and nERR@10.

We report the averaged immediate search accuracy for all sessions. It was worthy noting that session lengths vary. To average across sessions with different lengths, we made all sessions equals to the maximum session length in a dataset. TREC 2011 – 2014 Session have different maximum session lengths; they were 11 in TREC 2011 and 2012, 21 in TREC 2013 and 8 in TREC 2014. When a session was shorter than the maximum session length, we used the retrieval results from its own last iteration as the retrieval results for iterations beyond its own last iteration. In addition, since TREC did not provide any retrieval results for the

last query, the Original runs has no value at the last iteration.

Figures 8 to 11 plot the immediate search accuracy for TREC 2011 – 2014 Session Tracks averaged over all sessions. We observed that Win-Win search’s immediate nDCG@10 was statistically significantly better than the Original run’s immediate nDCG@10 almost at every iteration. In the worst case, the first iteration of TREC 2012, the nDCG@10 scores for the Win-Win run and the Original run were still roughly the same. For nERR@10, in TREC 2011 and 2012, the Win-Win run outperforms the Original run starting from the second iteration, while in TREC 2013 and 2014, it outperforms the Original run in every iterations. At the last iteration, Win-Win outperforms Original by a statistically significant 66.3% in nDCG@10 (p-value $\leq$.05, one-sided t-test) in TREC 2011, 57.6% in TREC 2012, 253% in TREC 2013, and 56.1% in TREC 2014.

Another interesting finding was that Win-Win search’s immediate search accuracy increases while the number of search iterations increases. In Figure 8, the nDCG@10 starts at 0.3860 at the first iteration and increases dramatically 36.1% to 0.5252 at the last iteration. We observed similar trends in Figure 9 to 11. It suggests that by involving more search iterations, i.e., learning from more interactions between the user and the search engine, Win-Win was able to monotonically improve its search accuracy.

This experiment investigated how legitimate the proposed states were in presenting the hidden mental states of users.

First, we used examples to demonstrate the state transitions in sessions. Session 87 was a long session with 21 queries. The chain of decision states identified for this session was : $S_{NRR}(q_1=\text{best us destination}) \rightarrow S_{RT}(q_2=\text{distance new york boston}) \rightarrow S_{NRT} \rightarrow S_{NRR} \rightarrow S_{NRR} \rightarrow S_{RR} \rightarrow S_{RR} \rightarrow S_{NRR} \rightarrow S_{RT} \rightarrow S_{RT} \rightarrow S_{RR}(q_{11}=\text{boston tourism}) \rightarrow S_{NRR}(q_{12}=\text{nyc tourism}) \rightarrow S_{NRR}(q_{13}=\text{philadelphia nyc distance}) \rightarrow S_{NRR} \rightarrow S_{RT} \rightarrow S_{NRR} \rightarrow S_{RR} \rightarrow S_{NRT} \rightarrow S_{NRT}(q_{19}=\text{philadelphia nyc travel}) \rightarrow S_{NRT}(q_{20}=\text{philadelphia nyc train}) \rightarrow S_{NRT}(q_{21}=\text{philadelphia nyc bus})$. The states correctly suggests that the user was in the exploration states (RR, NRR, NRR) from q_{11} to q_{13} , while he keeps changing queries to explore from city to city (boston, new york city, and philadelphia). The user eventually finds the cities, philadelphia and nyc, that fulfill the information need – “*best US destinations within a 150-mile radius*”. During the last 3 queries, the user exploits the current subtopic (philadelphia and nyc) to find out more specifics on transportations (travel, train, bus) about them. The system correctly recognizes the last three states as exploitation states (NRT, NRT, NRT). This example suggests that the proposed states were able to reflect the real user decision states quite accurately.

Second, we examined state transition patterns in long sessions since they contain enough transitions to study. Figure 12 plots state probabilities, state transi-

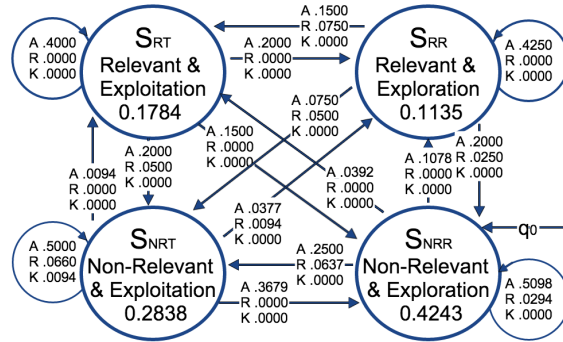


Figure 12: Long sessions (length greater than 4). Transition probabilities were listed with actions: Add (A), Remove (R), and Keep (K).

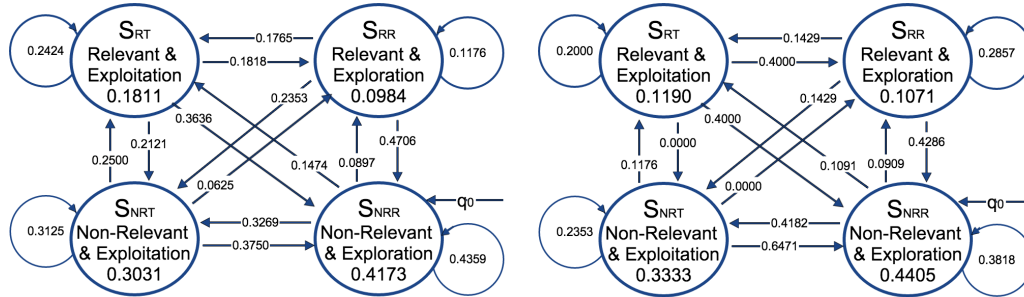


Figure 13: Factual and Specific sessions.

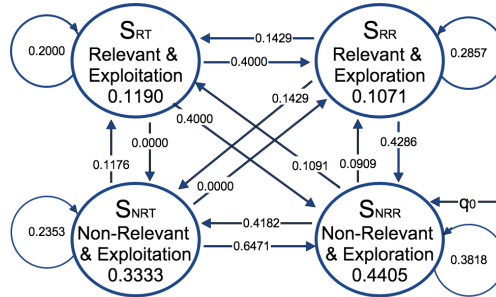


Figure 14: Factual and Amorphous sessions.

tion probabilities, and that under different user actions for long sessions (sessions with 4 or more queries). The data were combined from both TREC 2012 & 2013 Session Tracks. We noticed that NRR (non-relevant and exploration) was the most common state (42.4%). This reflects that a user spend may a long time to explore while receiving non-relevant documents. On the contrary, the RR state (relevant and exploration) was the least common state (11.3%). Moreover, we saw that state transitions were not uniformly distributed. For instance, the transition from NRT to both relevant states (RT and RR) were very rare (in total 5.65%). In addition, we noticed that actions were related to state transition probabilities. There were 90.8% transitions generated by adding terms and among all the transitions with removing terms, 84.8% of them lead to exploitation states (RT or NRT).

Third, we found that state probability distribution and state probability transitions differ among different session types. We plotted the state probabilities and transition probabilities in Figures 13 to 16 for four different TREC session

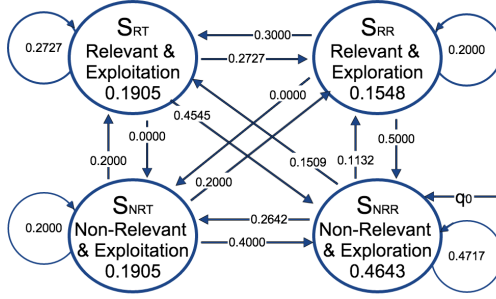


Figure 15: Intellectual and Specific sessions.

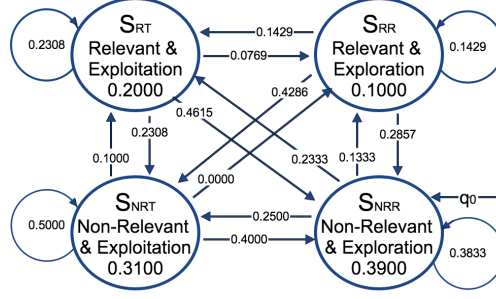


Figure 16: Intellectual and Amorphous sessions.

Table 11: Search effectiveness of the Bandits approach from Iteration 1 to Iteration 10 on TREC DD 2016 Ebola Dataset (* indicates stat. significant improvement over rmit-lm-psg-max, ($p < 0.05$, t-test, one-sided))

Algorithm	CT@1	ACT@1	CT@10	ACT@10
LinUCB	0.278	0.208	0.098*	0.135*
rmit-lm-psg-max	0.246	0.268	0.049	0.089
LDA_Indri73	0.260	0.187	0.069	0.115
TREC Median	0.260	0.187	0.076	0.119
TREC Best	0.319	0.236	0.176	0.150

types, which were created along two aspects: search target (*factual* or *intellectual*) and goal quality (*specific* or *amorphous*). Suggested by [53], the difficulty levels of the session types usually were FactualSpecific ; IntellectualSpecific ; FactualAmorphous ; IntellectualAmorphous. An interesting finding was that as the session difficult level increases, the transition probability from state NRT (non-relevant and exploitation) to state RT (relevant and exploitation) becomes lower: FactualSpecific (0.25), IntellectualSpecific (0.2), FactualAmorphous (0.12), IntellectualAmorphous(0.1). It suggests that the harder the task was , the greater the necessity to explore rather than to exploit, when the user was not satisfied with the current retrieval results. In addition, we observed that *Intellectual* sessions (Figures 15 and 16) have a larger probability, 0.1548, to be in the RR (relevant and exploration) state than the other session types (on average 0.1018).

We also compared our approach on the TREC Dynamic Domain Tracks. The TREC median and best scores were aggregated from the median and best score of all the runs at each iteration. We also compared our system with a submitted

Table 12: A comparison of the scores of system with manual search completion and automated search completion (* indicates stat. significant improvement of automated system over manual system, ($p < 0.05$, t-test, one-sided)).

Algorithm	1	3	5	7	10
Manual CT	0.277	0.139	0.090	0.068	0.049
Automated CT	0.277	0.149	0.116	0.104*	0.098*
Manual ACT	0.208	0.170	0.141	0.122	0.101
Automated ACT	0.208	0.172	0.153*	0.143*	0.135*

run called `rmit-lm-psg-max[2]` that uses a passage language model for document ranking, which represents a vastly different approach to the one presented in this research, and a submitted run, `LDA_Indri73` that uses MDP, a more similar approach to our system.

The results in Table 11 show that our bandits algorithm achieved significantly better CT and ACT scores compared to `rmit-lm-psg-max` by the tenth iteration. Our system also showed scores similar to the TREC median scores and `LDA_Indri73` from iteration one to iteration ten.

Our comparison with TREC best, however, showed that there is still much potential for our system to improve. The basic algorithms used for our actions and our use of the contextual bandit algorithm overall provides an effective method for retrieving relevant documents and selecting when to stop but leave room for continued improvement. As most of our actions use quite straightforward tactics for query re-formulation, it was clear there was large room to improve the search effectiveness if we employed more sophisticated query re-formulation tactics. Nonetheless, these comparisons show that using the context of the current iteration to select the next action in the search provides a good method for navigating through the search and finding relevant documents. These results also indicate that our approach was effective with even a simple set of actions.

To analyze the effects of automating the search completion, we ran our system with a manually set stopping condition instead of the “stop” action. Table 12 shows a comparison of that run with our automated system. Both systems begin with the same effectiveness, but the faster rate of decrease in the manual system demonstrates that our system was capable of detecting a good end to the search and provides a statistically significant improvement for our system’s search effectiveness.

The policy created by the bandit to make decisions on how to navigate through the search focused on retrieving as many relevant documents as possible. For ex-

Table 13: Example policy created by the bandit algorithm.

Action	Results
Add “Kaci Hickox actions” → “Kaci Hickox actions CDC”	3 relevant
Add “Kaci Hickox actions CDC” → “Kaci Hickox actions CDC obtain”	5 relevant
Weight “1.0 Kaci 1.0 Hickox 1.0 actions 1.0 CDC 1.0 obtain” → “1.0 Kaci 1.0 Hickox 1.0 actions 0.91 CDC 1.0 obtain”	4 relevant
Weight “1.0 Kaci 1.0 Hickox 1.0 actions 0.91 CDC 1.0 obtain” → “0.93 Kaci 0.96 Hickox 1.0 actions 0.91 CDC 1.0 obtain”	2 relevant
Stop	
Weight “1.0 natural 1.0 immunity” → “.87 natural .9 immunity”	1 relevant
Weight “.87 natural .9 immunity” → “.7 natural 1.0 immunity”	no relevant
Stop	

ample, Table 13 shows an example from a search on Kaci Hickox, who resisted a quarantine as she insisted she was not infected. First, the bandit selected “add”, and found that had good results. Specifically, the bandit system found that exploring a subtopic on her actions against the CDC led to many relevant documents. Then, when the bandit discovered a subtopic that seemed to be very relevant (with 5 relevant documents), it chose to continue the search that subtopic with a shift in focus through the action “weight,” also to good results. It finally ended the search when the effectiveness started to decrease.

During another search task, the bandit system tried using action weight to good results, turning “1.0 natural 1.0 immunity” to “.87 natural .9 immunity,” with 1 relevant document. It therefore continued to choose action weight again. However, the new query “.7 natural 1.0 immunity” retrieved no relevant documents. At this point, the bandit system chose to stop the search, deciding that there had been enough relevant documents retrieved and that the search was becoming ineffective. These examples demonstrate that the bandit understands the search path being explored and the search’s current effectiveness. It was able to discover the strategies that lead to relevant results, and chooses “stop” at points when the effectiveness of the search begins to drop off.

Point Fact: Ground Truth Data

Results Summary

Teams	Configuration	Mean Average Precision	Mean Point Fact Metric*
Georgetown	Georgetown Extractions	0.71	0.82
ISI	ISI Extractions	0.67	0.63
ISI	Lattice Extractions	0.38	0.25
ISI	ISI and Lattice Extractions	0.43	0.65
Uncharted	Uncharted Extractions	0.74	0.48
Uncharted	Uncharted and Lattice Extractions	0.78	0.47
Uncharted	Both Extractions and CMU Clusters	0.78	0.47

Figure 17: Memex Evaluation Results - Georgetown

Lastly, we participated in the Memex search evaluation in 2015-2016. Based on the above mentioned work on dynamic search, we developed a question answering system which considers the domain-specific search as a post-retrieval question answering (QA) research topic.

One of the major challenges in QA was to overcome the mismatch in the lexical representations between the query space and document space. This mismatch, also known as the QA gap, was caused by the differences in the set of terms used in the question formulation and answer strings in the corpus. Given a source, such as the QA corpus, that contains only a relatively small number of answers to a query, we were faced with the difficulty to map the questions to answers by way of uncovering the complex lexical, syntactic, or semantic relationships between the question and the answer strings.

We investigated the integration of both linguistic knowledge and external resources for question answering. In particular, we described a high performance question answering system and analyze its effectiveness using the Memex HT data benchmark. Our results show that combining lexical information and external resources with a custom text search produces an effective question-answering system.

Figure 17 show the official Memex Domain-Specific Search Evaluation results. The Georgetown team won the top position for the point fact questions.

In summary, the research presented a novel dynamic search framework, Win-Win search and its more efficient versions, that used a dual-agent stochastic game

```

<topic name="Return of Klimt paintings to Maria Altmann" id="dd17-1" num_of_subtopics="3">
  <subtopic name="Austrian Actions" id="106" num_of_passages="17">
  </subtopic>
  <subtopic name="Maria Altmann's legal actions" id="104" num_of_passages="14">
  </subtopic>
  <subtopic name="Sale after the return" id="393" num_of_passages="89">
  </subtopic>
</topic>

```

Figure 18: Sample Topic of TREC 2017 DD Track.

to model the interactions between user and search engine. With a careful design of states, actions, and observations, the new framework was able to perform efficient optimization over a finite discrete set of options. The experiments on TREC Session 2011 – 2014 datasets show that the proposed framework was highly effective for dynamic search.

Dynamic search is a complex IR task. The complexity comes from the involvement of many more factors other than just terms, queries and documents in most existing retrieval algorithms. The factors include query reformulations, clicks, time spent to examine the documents, personalization, query intent, feedback, etc. Most existing work on sessions and task-based search focuses on diving into one aspect. Through significantly simplifying the factors, we realized the integration of all the factors in a unified framework. For example, we simplified users' decision states into only four states, and discretize user actions and search engine actions into a finite number of options. Such simplifications were necessary in creating practical search systems. We hope our work calls for future adventures in the fields of RL in IR and game theory in IR.

3.2 Dynamic Search Evaluation

3.2.1 Text REtrieval Conference (TREC) Dynamic Domain Tracks 2015-2017

As part of our work in the DARPA Memex Program, the Georgetown Team led the effort in organizing a three-year dynamic search evaluation for special domains in the Text REtrieval Conference (TREC) conferences from 2015 to 2017. The evaluation Track was called Dynamic Domain (DD).

The goal of dynamic domain track was promoting the research of dynamic, exploratory search within complex information domains, where the search process was usually interactive and user's information need was also complex. Dynamic Domain (DD) track has been held in the past three years. This track's name includes two parts. "Dynamic" means the search process may contain multiple runs of iteration, and the participating system was expected to adapt its search

```

<subtopic name="Sale after the return" id="393" num_of_passages="89">
  <passage id="3656">
    <docno>1752374</docno>
    <rating>2</rating>
    <text><![CDATA[In yet another unexpected turn, Austria in February declined an option to buy the
    paintings from the Altmanns, for reasons that remain unclear.]]></text>
    <type>MANUAL</type>
  </passage>
  <passage id="3657">
    <docno>1770282</docno>
    <rating>3</rating>
    <text><![CDATA[A dazzling gold-flecked 1907 portrait by Gustav Klimt has been purchased for the
    Neue Galerie in Manhattan by the cosmetics magnate Ronald S. Lauder for $135 million, the
    highest sum ever paid for a painting.]]></text>
    <type>MANUAL</type>
  </passage>
  <passage id="3658">
    <docno>1770326</docno>
    <rating>3</rating>
    <text><![CDATA[A dazzling gold-flecked 1907 portrait by Gustav Klimt has been purchased for the
    Neue Galerie in Manhattan by the cosmetics magnate Ronald S. Lauder for $135 million the highest
    sum ever paid]]></text>
    <type>MATCHED</type>
    <score>1</score>
  </passage>

```

Figure 19: Sample Relevance Judgement of TREC 2017 DD Track.

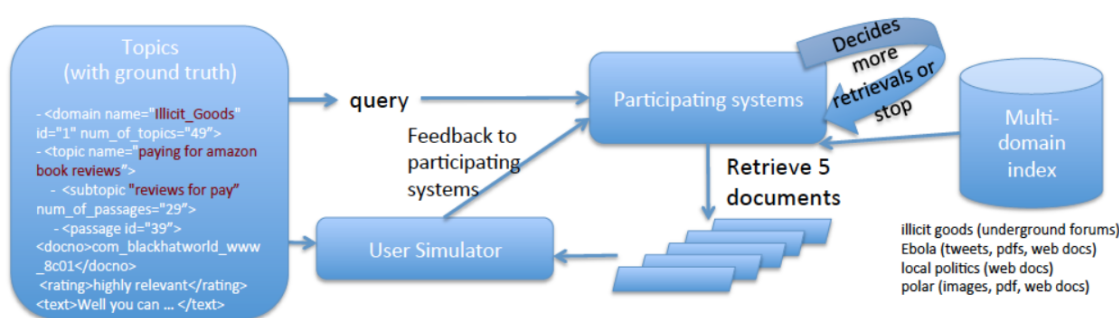


Figure 20: TREC Dynamic Domain (DD) Track Task Illustration.

algorithm based on the relevance feedback. “Domain” means the search task focuses on special domains, where user’s information need consists of multiple aspects, and the participating system was expected to help the user explore the domain through rich interaction. This task has received great attention and this track was inspired by interested groups in government, including DARPA MEMEX program. The settings of DD track were motivated by professional search, such as prior art patent search or criminal network finding, where rich interaction was a great asset for improving the search results and users require stringent relevancy on the documents returned.

In order to simulate the interaction between the search engine and the user, as well as to evaluate the whole search process, a simulated user (called Jig) was developed. During each step in interaction, participating system sends a list of

```

{
  "ranking_score": "833.00",
  "subtopics": [
    {
      "subtopic_id": "DD16-1.1",
      "passage_text": "Marine Lt. Col Doug Woodhams U.S. Army Africa Sgt. Bromley and Liberian Arme",
      "rating": 2
    },
    { ... },
  ],
  "doc_id": "ebola-45b78e7ce50b94276a8d46cfe23e0abbcbcd606a2841d1ac6e44e263eaf94a93",
  "topic_id": "DD16-1",
  "on_topic": "1"
}
{ ... }

```

Figure 21: Sample Feedback from the Simulated User (the Jig).

documents to the simulated user, and the simulated user returns a real-time feedback to the participating system. Participating system learns the real intention behind the search topic, adapts its search algorithm and generates the next list of documents. This process was repeated until the participating system believes the user's information need has been satisfied. All the documents returned by the participating system were saved for the evaluation of the whole search session.

The DD Track also uses fine-grained judgements. Different from open domain web search, all the relevance judgements were on the passage level, which expresses user's information need more accurately. Correspondingly, DD track also uses sophisticated metrics to evaluate the search results, which includes Cube Test, session-DCG and Expected Utility. All these metrics evaluated the whole search process and each provides a distinct view on the effectiveness and efficiency of the participating systems. In 2017, the DD track focused on the exploration of New York Times archives.

The task of TREC DD track was based on a concept that search was driven by the feedback instead of queries. Figure 20 illustrates the task. That was, a good search system need to learn user's real intents through the feedback given by the user regarding previous returned documents and the user does not need to reformulate queries to express or refine his/her information need. The search system needs to adjust its search algorithm so as to help the user explore the complex domain. Every search topic in DD track contains multiple aspects, which were referred as subtopics in this track. All the relevance judgements were on passage level. Every document may contain several relevant passages, each related to a subtopic with different relevance scores.

In the beginning, the search system receives an initial query (the topic name) indicating user's intention. Then, the search system retrieves five documents from

the index and sends them back to the user simulator Jig (Figure 38). Jig returns feedback about the returned five documents. The feedback sent back by Jig gives a detailed description about the relevance of returned documents on subtopic level. The search system needs to decide if it would continue returning documents or stop the current search. The search system may also consider how to rerank the documents so as to better satisfy user's information need.

This track was expecting participating systems achieve two basic points. First, the search system was expected to adapt its search algorithm based on the relevance feedback, for which rich information was provided in the feedback. Second, it was the search system's job, instead of the user's, to decide whether to continue search. Search results were evaluated using different sophisticated metrics, which measures the total amount of information the search system gains and the effort of user from various points of view.

In summary, the dynamic domain track has been running in the past three years at TREC and this was the final year. DD track always focuses on interactive and exploratory search task with the vision that a good search system should be a guide to discover the domain of user's interest by adaptively learning user's intention. Participants have tried various methods marching towards this goal. With the rising of Artificial Intelligence (AI) in recent years, it would be exciting to see that this track shares a very similar goal with many AI tasks, that was , building an intelligent system that understands human's mind better.

3.2.2 Annotating and Evaluating Complex Search Processes

Topics were developed by six NIST assessors in over six weeks during the summer of 2017. A topic, which was like a query, was the main search target for the whole search process. Every topic contains several subtopics, each addresses one aspect of the search topic. Each subtopic contains several number of passages which was discovered from the entire corpus. Each passage was graded based on the relevance between its content and the subtopic. An annotation tool was developed to help the assessors find the complete set of passages that were relevant to the query. Also, near-duplicate detection was utilized to help find possible relevant passages that may be missed. The graded passages were treated as the complete set of judgement.

Four algorithms were provided for search in topic level and two algorithms were used for search in subtopic level. Assessors first conduct search in topic level. They can then go into the detailed page of every document where they would give rich feedback to the annotation tool. They can decide if a document was irrelevant or duplicated. They can also drag and drop relevant passages to the subtopic box on the right side. Each passage was then graded based on the extent of relevance.

The highest relevance score was 4 for key results and the lowest relevance score was 1 for marginally relevant.

In the topic level search, three mainstream open source search engine, Lemur, Solr and Terrier, and an active learning search algorithm were provided. DD track expects the combination of different search engines can reduce the inherent bias of each individual one. The active learning algorithm uses the feedback, i.e. the documents graded by assessors before, to refine its search results. For every query, assessors were required to search in every one of the four search buttons and go over the ranking lists so as to cover as many relevant documents as possible.

In the subtopic level search, two search algorithms were used to help assessors find more relevant passages. One of them uses the passages that have been tagged and the subtopic name to search for relevant documents, the other one only uses tagged passages to search. For every subtopic, assessors were also required to search in each one of them so as to find a complete set of relevant passages.

After completing the human annotation, near-duplicate detection was used to find passages that were actually relevant but may be missed by assessors. In the ground truth data, passages that were tagged with “MANUAL” were those discovered by human assessors while passages that were tagged with “MATCHED” were those discovered using detection algorithm, which was based on its similarity to the human annotated passages.

In order to provide a comprehensive evaluation of the whole search process, in 2017, DD track uses several evaluation metrics to measure the performance of search systems from different perspectives. The primary metric used in DD track was Cube Test. DD track also uses session-DCG and Expected Utility for evaluation. These metrics reveal different aspects of the dynamic search process.

Cube Test (CT) was a search effectiveness measurement evaluating the speed of gaining relevant information (could be documents or passages) in a dynamic search process. It measures the amount of relevant information a system could gather and the time needed in the entire search process. The higher the Cube Test score, the better the IR system. It was defined as:

$$CT = \left(\sum_{i=1}^L \sum_{j=1}^{|list_i|} \sum_c \theta_c * rel_c(i, j) * \gamma^{n(c, i, j-1)} * I \left(\sum_{m=1}^{i-1} \sum_{n=1}^{|list_m|} rel_c(m, n) + \sum_{n=1}^{j-1} rel_c(i, n) < MaxHeight \right) \right) / L \quad (40)$$

where L was the number of iterations so far, $|list_i|$ was the number of documents returned at the i^{th} iteration. c was a subtopic, c was the importance factor of

subtopic c , $rel_c(i, j)$ was the relevance score of the j^{th} document that was returned in i^{th} iteration regarding subtopic c , $n(c, i, j - 1)$ was the number of relevant documents found on subtopic c before the j^{th} document in the i^{th} iteration, was the discounting factor, $I(*)$ was an indicator function and $MaxHeight$ was the cap for each subtopic.

Session-DCG (sDCG) extends the classic DCG to a search session which consists of multiple iterations. The relevance scores of results that were ranked lower or returned in later iterations get more discounts. The discounted cumulative relevance score was the final result of this metric. It was defined as:

$$sDCG = \sum_{i=1}^L \sum_{j=1} |list_i| \frac{rel(i, j)}{(1 + \log_b j) * (1 + \log_{bq} i)} \quad (41)$$

sDCG does not consider subtopics so $rel(i, j)$ was the relevance score that was accumulated over all the subtopics. Both b and bq were discounting factors.

Expected Utility (EU) scores different runs by measuring the relevant information a system found and the length of documents. The relevance scores of documents were discounted based on ranking order and novelty. The document length was discounted only based on ranking position. The difference between the cumulative relevance score and the aggregated document length was the final score of each run. It was defined as:

$$EU = \sum_{\omega} P(\omega) \left(\sum_{(i,j) \in \omega} \left(\sum_{c \in d_{i,j}} \theta_c * \gamma^{n(c,i,j-1)} \right) - a * len(i, j) \right) \quad (42)$$

EU assumes that the user only reviews a subset (ω) of documents returned. $P(\omega)$ was the probability of subset being reviewed, $len(i, j)$ was the length of document that was ranked j^{th} in the i^{th} iteration and a was the coefficient.

The parameters used for DD track evaluation were as follows: In CT, $\gamma=0.5$ and $MaxHeight=5$. In sDCG, $b=2$ and $bq=4$. In EU, $\gamma=0.5$ and $a=0.01$. All the subtopics within the same topic were assumed to be equally important.

3.2.3 Finding Bounds for the Evaluation Metrics

Dynamic search was a complex Information Retrieval (IR) task. As a result, its evaluation was also complex. A great number of factors need to be considered in the evaluation of session search. They include document relevance, document novelty, aspect-related novelty discounting, and user's efforts in examining the documents. Due to increased complexity, most existing session search evaluation metrics were NP-hard. Consequently, the optimal value, i.e. the upper bound, of a metric highly varies with the actual search topics. In Cranfield-like settings such as

the TREC Conferences, scores for systems were usually averaged across all search topics. With undetermined upper bound values, however, it could be unfair to compare IR systems across different topics. This research addresses the problem by investigating the actual per topic upper bounds of existing session search metrics. Through decomposing the metrics, we derived the upper bounds via mathematical optimization. We showed that after being normalized by the bounds, the NP-hard session search metrics were then able to provide robust comparison across various search topics.

Most metrics measure the amount of information a user gains during a search session. The gain was usually represented as the sum of relevance of the documents that have been returned so far. The relevance scores were usually assigned by third party annotators as ground truth scores. They can be graded or binary scores. Generally, as shown in previous research, it was believed that there was a positive correlation between a third party annotated relevance score and a user's satisfaction level [1, 29]. In this research, instead of a user study, we would mainly focus on discussing evaluation in the Cranfield-like settings which rely on third party annotations.

Discounting document relevance was a common technique used in IR evaluation. There were two main types of discounting methods. First, discounting was done based on ranking orders. The idea was that the relevance scores of the lower ranked documents were discounted (as in DCG [33]) because it was assumed that users were less likely to read those documents hence less gains come from them. Second, discounting was done based on content redundancy. The relevance scores of documents that repeat on topics that have appeared in earlier documents were discounted (as in α -nDCG [18]), too.

Some IR evaluation metrics also take into account users' efforts (sometimes also known as *cost*). The cost can be regarded as a user's effort, both mentally and/or physically, spent in the session. It was usually simplified as the time spent in completing the entire task/session [54, 69], or the aggregated lengths of documents a user has read [82]. As more user efforts were put into search activities, the user's satisfaction level was expected to decrease. Kokubu et al. has found a positive correlation between user satisfaction and the reciprocal rank of relevant results in QA systems [46], which implies that a user was more satisfied if less documents were needed to examine in order to find the answers. The cost of a document may be discounted as well (as in Expected Utility [82]).

Most session search metrics handle all of the above factors and usually combine them into a single formula. The complexity of these metrics was quite high and most of them were NP-Hard. Consequently, the optimal value, i.e. the upper bound, of a metric highly varies with the actual search topics. That was to say, the bounds were not only decided by the mathematical definition of each metric,

Table 14: Document Relevance Scores in Ground Truth

topic	document
1 (2 subtopics)	d1(1.1, 1) d2(1.2, 3)
2 (4 subtopics)	d1(2.1, 4) d2(2.2, 4) d3(2.2, 2) d4(2.3, 4) d5(2.4, 4)

Table 15: The Importance of Normalization

systems	CT-topic 1	CT-topic 2	CT avg	normalized-CT avg
optimal	4	17	/	/
system 1	1	16	8.5	0.596
system 2	3	14	8.5	0.787

but also affected by the ground truth data of each topic. For instance, some search topics might be easy (with high optimal value for those topics) because many relevant documents were available and many search systems can achieve pretty high evaluation metric scores on those topics; while other topics might just be difficult for all search systems (with low optimal values for those topics). In Cranfield-like settings such as the Text REtrieval Conference (TREC), however, evaluation metric scores for systems were usually averaged across all search topics without considering the per topic upper bound a metric could achieve. Neglecting the differences in those bounds could be unfair when evaluating the systems across different topics.

A toy example was shown in Tables 14 and 15. Suppose two systems were evaluated on two topics with two and four subtopics respectively. In Table 14, $d1(1.1, 1)$ means document $d1$ was relevant to subtopic 1.1 with a relevance score of 1 (scaled from 1 to 5). Assume that each system returns five documents. *System 1* found $d1$ for *topic 1* and $d1, d3, d4, d5$ for *topic 2*; *System 2* found $d2$ for *topic 1* and $d1, d2, d4, d5$ for *topic 2*. Other documents returned were irrelevant. Suppose that Cube Test [54] was used to evaluate session search effectiveness. We also assumed that equal amount of time was needed to read each document and the discounting factor was 0.5. The optimal scores and the actual scores were shown in Table 15. As we can see here, the two systems' averaged CT scores were the same (8.5) thus it was hard to tell which system was better. However, the raw CT scores do not reflect the actual performance of the two systems. *Topic 2* has a high upper bound (17), which suggests an easy search topic. Therefore, the higher CT score from *system 1* (16) does not support that it was a better system than *system 2* which does an impressive job on the more difficult topic (*topic 1* with an upper bound of 4). The issue can be resolved by proper normalizing the raw

Algorithm 3 optimal score on a single ranking list

Input: $|list_i|$ where $i = 1, 2 \dots L$,raw (relevance/cost) scores: r_k of document d_k , $k = 1, 2, \dots, n$,discount function: $discount(i, j)$ optimization direction: $IsMaximize = true$ or $false$ **Output:** optimal score on a single ranked list $POS = \{(1, 1), \dots, (1, |list_1|), (2, 1), \dots, (L, |list_L|)\}$

// Set of all possible document ranking positions

 $SP = Queue(sort(POS) \text{ by } discount(i, j) \text{ in ascending order})$ **if** $wasMaximize$ **then** $D = Queue(sort(d_k) \text{ by } r_k \text{ in ascending order})$ **else** $D = Queue(sort(d_k) \text{ by } r_k \text{ in descending order})$ $opt = 0$ **while** SP was not empty and D was not empty **do** $(i, j) = SP.dequeue()$ $d_k = D.dequeue()$ $opt += r_k * discount(i, j)$ **return** opt

scores by the per topic upper bounds, as shown in the last column in Table 15. The example demonstrates that knowing the per topic metric bounds was very important in fairly evaluating IR systems.

In this work, we investigated existing session search evaluation metrics and focus on the following research questions: *What was the best possible optimal metric value that a system could achieve?* To answer this question, we first took apart existing session search metrics into components. We then analyzed the rationale behind each component and the ways used to combine them. Based on the component-based analysis, we computed the optimal score or the bounds of these metrics, which were then used for score normalization later. The new normalized metrics were experimented on the official runs submitted to the TREC 2016 DD Track. Our results show that these NP-hard session search metrics were then able to provide robust comparison across different topics.

Greedy algorithms have been proposed to approximate the optimal score in novelty-related metrics [13, 18], where multiple ranked lists were required to be optimized simultaneously. However, the focus of a greedy algorithm was to produce an ideal document list, which was not required for score normalization. It was also shown that the results yielded by a greedy algorithm can be far below the optimal score on certain topics [13]. From this point of view, optimization based on

rearrangement inequality was more proper because it was able to lead to a tighter bound and was very efficient especially when the number of relevant documents was large.

Our optimization algorithm was shown in Algorithm 3. It handles both minimization and maximization based on different settings. In the algorithm, all the possible slots were first ranked based on the discount value it would receive. When maximization was needed, a document with higher raw score would be put at a position with higher $discount(i, j)$ value. When minimization was needed, the document with higher raw score would be assigned to a position with lower $discount(i, j)$ value. The algorithm forms the basis for optimizing of the dynamic search metrics.

Session DCG (sDCG) was essentially discounted cumulated gain for a search session. Computing the optimal sDCG score can be expressed as

$$maximize \sum_{i=1}^L \sum_{j=1}^{|list_i|} \frac{rel(i, j)}{(1 + \log_b j) * (1 + \log_{bq} i)} \quad (43)$$

Since subtopics were not considered in sDCG, only one document ranking list needs to be optimized. Algorithm 3 can be directly used by setting $IsMaximize = true$ and $discount(i, j) = \frac{1}{(1 + \log_b j) * (1 + \log_{bq} i)}$.

Different from the normalization method proposed in [34], there was no duplicated result in our ideal ranked lists. Our optimal sDCG score represents the optimal performance a system can achieve in the session. The range of sDCG score of any system on a topic was $[0, opt]$ when the session length L was fixed.

Computing the optimal score of CubeTest was not as intuitive as that of sDCG. In fact, computing the optimal score of Cube Test was NP-Hard even for a special case of CT, which was defined as

$$maximize CT = \frac{\sum_c \sum_d novelty_discount_{d,c} * gain_{d,c}}{\sum_d cost_d} \quad (44)$$

where $cost_d = 1$, $novelty_discount_{d,c}$ was boolean and was set to 1 iff. document d was the first relevant document found on subtopic c . In the special case, the discounted cumulated gain can be considered as the number of subtopics found by the IR system and the aggregated cost was the number of documents returned. The optimal CT score in this special case should have the highest number of subtopics found with the minimum number of documents returned.

The optimization problem can be transformed into the *Minimum Edge Dominating Set* problem in graph theory, which was NP-Hard [83]. An *edge dominating set* was a subset of edges satisfying the property that every edge that was not in

this subset was adjacent to at least one edge in this subset. The *minimum edge dominating set* was one such subset of edges that has the smallest size.

Here each document was considered as an edge and each subtopic was considered as a vertex. Then, documents (edges) in the *minimum edge dominating set* were the ones that need to be returned so as to achieve the optimal performance in the special form of CT. Because the special CT was NP-Hard, computing the optimal value of general CT was also NP-Hard. Nonetheless, CT's upper bound can still be computed based on *rearrangement inequality*. The optimal CT can be achieved by

$$\begin{aligned}
& \text{maximize} \sum_{i=1}^L \sum_{j=1}^{|list_i|} rel_c(i, j) * \gamma^{\sum_{l=1}^{i-1} |list_l| + (j-1)} \quad \forall c \\
& \text{minimize} \sum_{i=1}^L \sum_{j=1}^{|list_i|} cost(i, j)
\end{aligned} \tag{45}$$

Formula 45 requires to maximize a number of objective functions and minimize one objective function. The number of objective functions need to be maximized equals to the number of subtopics, i.e., $\#(c)$. The upper bound of CT can be derived by optimizing each of these target functions independently using Algorithm 3 and then combine them. For the maximization target functions, the input contains the raw relevance scores of all the documents regarding a given subtopic, and $discount(i, j) = \gamma^{\sum_{l=1}^{i-1} |list_l| + (j-1)}$. For the minimization target function, the input contains the raw cost scores of all the documents, and $discount(i, j) = 1$. The approximated upper bound may not be feasible in real situations. However, it provides good approximations of the real bound.

EU assumes that the user would scan the returned document in a top-down fashion with a probability p of stopping at some document in the current iteration. Once the user stops reviewing, (s)he would start the next search iteration. In order to make the computation tractable, Yang et al. [82] approximated the computation of EU by formula 46.

$$EU = \frac{1}{1-\gamma} \left(\sum_c \theta_c \left(1 - \gamma^{\sum_{\omega} P(\omega) n(c, \omega)} \right) \right) - a \sum_{\omega} P(\omega) len(\omega) \tag{46}$$

where $n(c, \omega)$ was the number of appearances of nugget c in the reviewed document subset ω and $len(\omega)$ was the total length of documents in ω .

The computation of $\sum_{\omega} P(\omega) n(c, \omega)$ and $\sum_{\omega} P(\omega) len(\omega)$ can be further expanded. $\sum_{\omega} P(\omega) n(c, \omega) = \sum_{i=1}^L \sum_{h_i=1}^{|list_i|} P(h_i) \sum_{j=1}^{h_i} rel_c(i, j)$
 $= \sum_{i=1}^L \sum_{j=1}^{|list_i|} rel_c(i, j) \sum_{h_i=j}^{|list_i|} P(h_i) \sum_{i=1}^L \sum_{j=1}^{|list_i|} rel_c(i, j) (1-p)^{j-1}$, where h_i was

the position where the user stops in the i^{th} iteration of the session. Similar transformation can also be applied to $\sum_{\omega} P(\omega) len(\omega) = \sum_{i=1}^L \sum_{j=1}^{|list_i|} cost(i, j)(1 - p)^{j-1}$. In EU, each nugget c has its own graded importance score θ_c . The relevance between a document and a nugget $rel_c(i, j)$ was binary (a document does or does not contain a nugget). The cost of a document, $cost(i, j)$, was its length.

The maximization of formula 46 was also NP-Hard. It was achieved by

$$\begin{aligned} & \text{maximize} \sum_{i=1}^L \sum_{j=1}^{|list_i|} rel_c(i, j) * (1 - p)^{j-1} \quad \forall c \\ & \text{minimize} \sum_{i=1}^L \sum_{j=1}^{|list_i|} cost(i, j) * (1 - p)^{j-1} \end{aligned} \quad (47)$$

Using similar methods as in CT, the upper bound of EU can be obtained by maximizing $\#(c)$ target functions and minimizing 1 target function independently with Algorithm 3. Note that the lower bound of EU was negative. It was because an IR system may return documents that were all irrelevant but still requires user's effort to read them, which leads to a negative EU score. The lower bound of EU can be derived by

$$\text{maximize} \sum_{i=1}^L \sum_{j=1}^{|list_i|} cost(i, j)(1 - p)^{j-1} \quad (48)$$

which can also be approximated using Algorithm 3.

In summary, dynamic search brings rich interactions between the user and the search system. The evaluation of session search encompasses many factors in search, such as relevance, novelty, and user's effort. As more factors were included into the evaluation, many metrics for session search evaluation become NP-Hard. In this research, we deconstructed those metrics, compute the optimal scores and use them for score normalization. Through experimenting on the TREC DD Track, we observed that (i) the variation of bounds of session search metrics on different topics was big and cannot be ignored; (ii) Using the bounds for normalization of those complex session metrics can bring in more fairness into evaluating across topics and yield more reliable evaluation. In addition, the upper bounds of the metrics could potentially be used as optimizing criteria for search systems to decide when to stop the search. Suppose a system has completed k iterations, by giving the upper bound of the first $k + 1$ iterations and computing its actual score in the first k iterations, the system could potentially make a better judgment on whether to continue or stop searching.

Table 16: Topic sample in TREC-DD 2016

Topic/Subtopic id	Topic/Subtopic name
DD16-1	US Military Crisis Response
– DD16-1.1	West African mission
– DD16-1.2	Key Personnel
– DD16-1.3	Personnel safety protocols

3.2.4 Results and Discussion

We experimented on the dataset and the official runs submitted to the TREC 2016 Dynamic Domain (DD) Track [78]. The dataset contains documents in various formats, including html pages and tweets. Each search topic contains several subtopics addressing different aspects of the topic. Every document may be relevant to multiple subtopics with different relevant grades. Subtopics within a topic were assigned identical weights. In total, there were 53 topics and 242 subtopics, with an average of 4.57 subtopics per topic. The ground truth data was created by NIST assessors. It contains 14,597 relevant documents, with an average of 291.47 relevant documents per topic⁵. Table 16 shows an example topic from TREC 2016 DD Track.

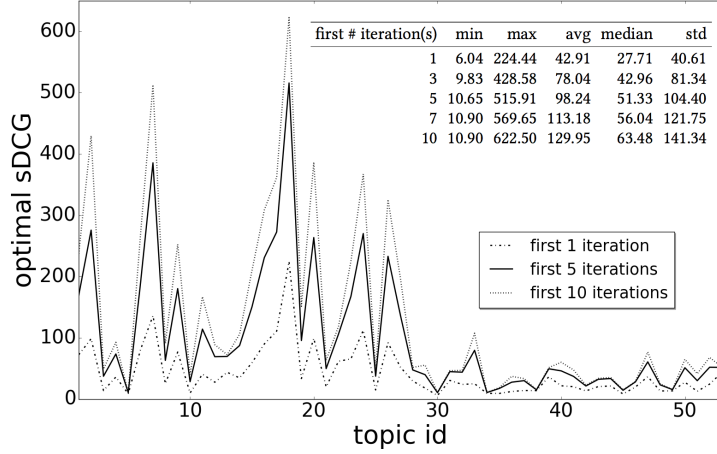
We conducted two sets of experiments. The first mainly studies the upper bounds (or the bound size, if the lower bound of the metric was negative) of the session search metrics. The second studies the influence of score normalization.

Figures 22(a), 22(b) and 22(c) plot the upper bounds/ bound sizes on different topics for sDCG, Cube Test and Expected Utility respectively at different session lengths. Within each figure, we also showed the statistics for the corresponding metric in a table. The statistics include the min, max, average, median and standard deviation of the bounds. The runs we studied were the 21 official runs submitted to TREC 2016 DD Track.

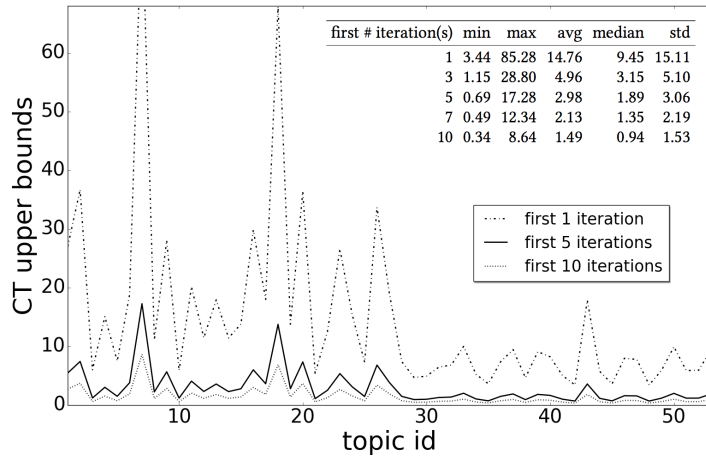
We observed that differences among the bounds of a metric across different topics were huge and non-negligible for all three session search metrics. This conclusion was true regardless of how many search iterations having been conducted. It suggests that without proper score normalization, it would be unfair to compare across different topics. A metric score averaged for all topics would be biased towards topics that have higher bounds.

For a particular search topic, as more search iterations were conducted, we observed that the metrics' upper bound/bound size changes. They change differently

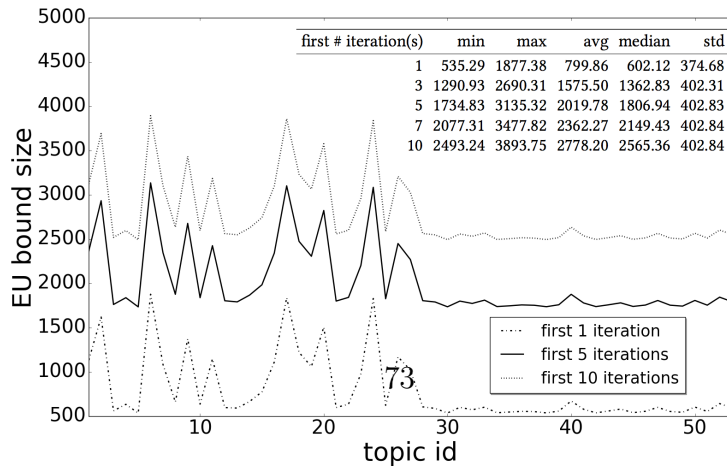
⁵Some documents were relevant to multiple topics



(a) optimal sDCG on TREC16-DD topics



(b) CT upper bounds on TREC16-DD topics



(c) EU bound size on TREC16-DD topics

Approved for Public Release; Distribution Unlimited.

Figure 22: Upper bounds/ bound sizes on TREC16-DD topics

for the three metrics. The optimal sDCG score increases as the search goes on with more iterations. It was because sDCG was essentially a gain function and it ignores the cost of search. Therefore, more search iterations would always increase chances of getting more relevant documents. On the other hand, the upper bound of CT decreases as the search keeps going. It suggests that the rate of gaining relevant information was decreasing as more iterations were used. It makes sense since once a user has learned relevant information in the initial runs, (s)he would not be too surprised for more relevant information at the later iterations thus the gains from those later iterations decrease. However, this observation might also only be related to the TREC 2016 DD dataset and tasks, where the search topics were mostly factual and informational. For navigational or learning-intensive search topics, we might be able to observe a different learning rate. Lastly, we observed that the bound size of EU enlarges as the session length increases. It comes from the fact that EU was a "net gain" function, more iterations mean more possible gain as well as more possible cost.

Moreover, as more iterations conducted in a session, the differences of the metric bounds across topics also change in various ways. For sDCG, the differences among topics become bigger. As a result, the optimal sDCG on different topics become more polarized as the number of iterations increases. For CT, this difference reduces a bit which indicates the system performance gets similar among the topics when more iterations were used. But the differences were still huge and cannot be neglected. For EU, the differences among topics remain relatively the same as the session develops.

Regardless of the changes on the upper bounds/bound sizes as more iterations were conducted, the difference of the optimal value a metric would produce for different topics was large and should not be ignored. We recommend taking per topic bounds into account for fairer evaluation.

Twenty-one runs from six groups were submitted to TREC 2016 DD Track. We selected six representative runs, one from each team, to examine the effects of score normalization. Figures 23, 24 and 25 plot the raw scores without and with normalization. The scores were averaged across all topics for selected runs.

Figure 23 compares the raw (left) and the normalized (right) sDCG scores. The raw sDCG scores were non-decreasing as the number of iterations increases. It was because sDCG was essentially cumulated gains. However, the normalized sDCG scores could increase, decrease or remain as a constant as the number of iterations increases. It suggests that some systems actually have been close to the optimal sDCG score at certain points whereas others do not perform that well. We noticed that systems such as *LDA_Indri73* and *UL_LDA_NE*, whose raw sDCG values increase (non-decrease) in Figure 23 (left) but decreases after being normalized (Figure 23 (right)). It suggests that even though the raw gains keep

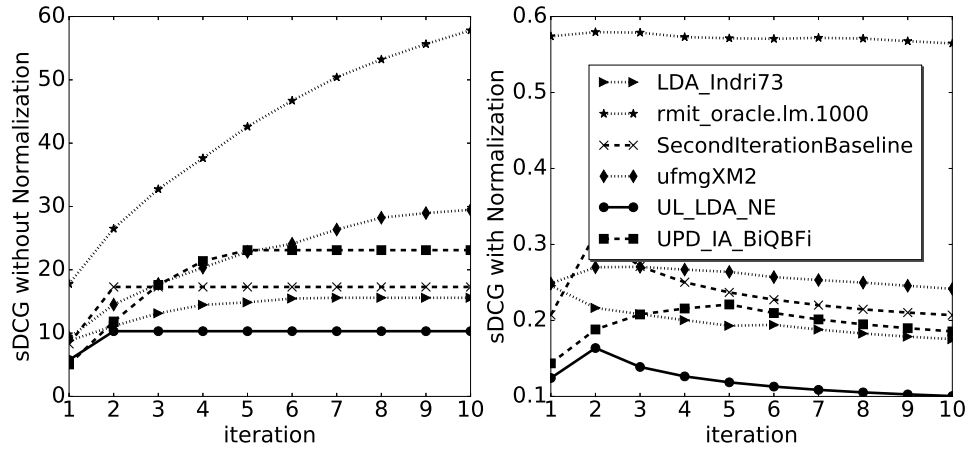


Figure 23: sDCG on TREC16-DD

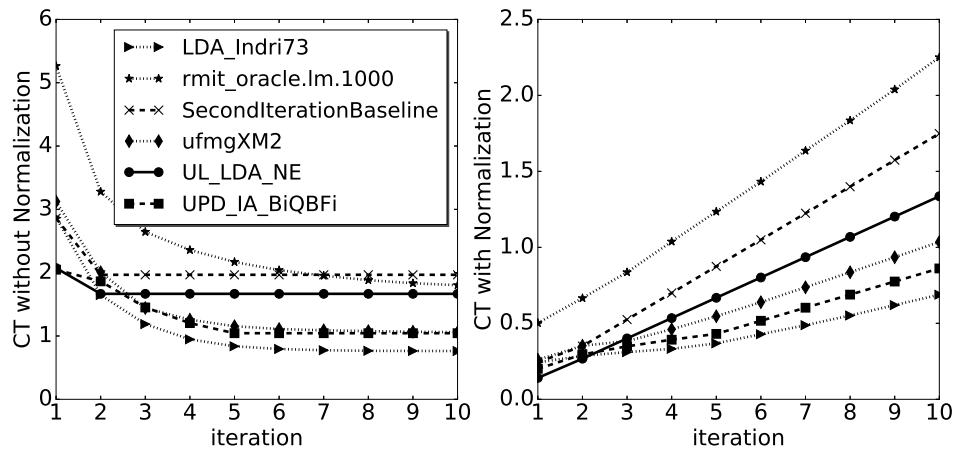
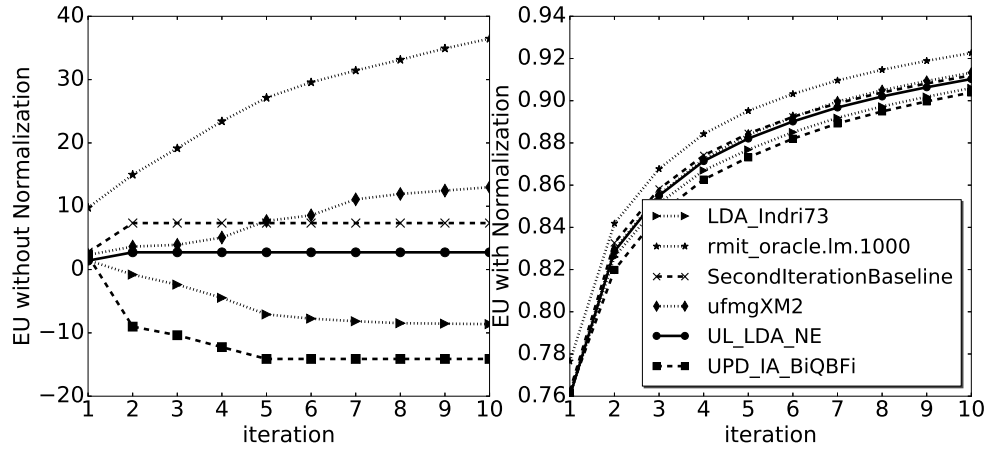
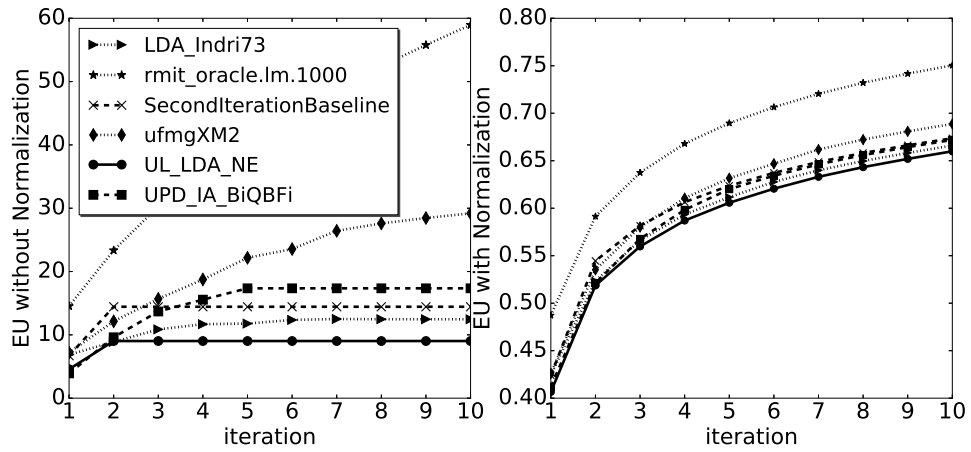


Figure 24: CT on TREC16-DD



(a) $a = 0.001$



(b) $a = 0.0001$

Figure 25: EU on TREC16-DD

increasing, they increase at a much slower rate than that of the optimal sDCG increases.

Figure 24 compares the raw (left) and the normalized (right) CT scores. Most raw CT scores were decreasing, suggesting that as more iterations were involved, the rate of getting relevant information by those systems reduces. Some raw CT scores remain the same because their systems stop the search after a certain number of iterations. Based on the definition of CT, after a system stops, its gain and cost in the following iterations were all zero therefore its CT score won't be affected. On the other hand, all the normalized CT scores increase as the session develops. It was yielded from the sharply declining upper bounds of CT. We even saw some normalized CT scores increase so much that they become greater than 1. The normalized CT scores actually imply that, in session search, choosing the right time to stop the search can help an IR systems maintain its efficiency so as to improve users' satisfaction.

Figure 25 compares the raw (left) and the normalized (right) EU scores. As the number of iterations increases, the raw EU scores may increase (Figure 25(a)) or decrease (Figure 25(b)) while the normalized EU scores always increase. We realized that whether the raw EU score would increase or decrease was highly influenced by the choice of the parameter a . With different parameter settings, the raw EU curves for the same run could be completely different. It confirms our conclusion that it was not appropriate to directly add or subtract gain by cost. On the other hand, the shape of the normalized EU score was not sensitive with different parameter settings. As a result, the normalization based on the metric bounds yields a more robust metric. Meanwhile, the increase of normalized EU score suggests an enlarged gap between the raw score and the metric's lower bound while the change on the difference between the raw score and upper bound may vary. Nonetheless, the enlarged gap suggests that all the systems have at least moved away from the worst case (the lower bounds).

3.3 Authorship Detection

In the DARPA Memex program, we worked on many domain-specific cybercrime datasets. The Cybercrime dataset contains a large volume of anonymous texts written by criminals. Grouping these documents by authorship would be helpful in finding the criminals' identity. Typically, authorship clustering, which was grouping texts by authorship, provides solutions to such questions. It was an unsupervised machine learning problem, and the number of authors was unknown. However, in our study, the texts were often in paragraph size, and most likely contain spelling errors or lack sentence structures. Extracting syntactic or semantic features can be very challenging, and the expected performance of that was also

contain the location where the criminals were looking for their potential customers, another advertisement may include the author's contact method. These clues were more useful when they were grouped together.

Authorship clustering is able to provide solutions for such tasks. The task of authorship clustering was that given a corpus with N documents written by an unknown number of authors, we grouped a part of them into k clusters where each cluster represents an unique author. Previous studies has mainly selected n -grams at character and word levels as features in clustering, such as [23], [22], and [45]. However, while current authorship clustering studies focused on literary texts in most cases, the advertisements in our corpus were usually short and in paragraph size, and often lack complete sentence structures. Many of them even contain spelling errors, which makes n -grams methods perform not well in our study.

Some other studies required the true number of total authors to be known before the actual clustering step, such as [49] and [32]. However, in our study, the true number of total authors remain unknown through the whole process.

There also had been studies using linguistic analysis such as POS n -grams [19] and sentence to sentence sequencing [3]. Due to the fact that the advertisements in our corpus often lack complete sentence structures, linguistic analysis including semantic and syntactic analysis may not be applicable. Sample text in Figure 26 demonstrates spelling errors (red highlighted) and the lack of complete sentence structures (blue highlighted).

We proposed an authorship clustering approach using information retrieval techniques as a "pre-step". In each iteration, cluster centroid document was randomly selected from document bins, which were divided by documents' word number. Candidate documents were retrieved with the "fingerprint query" of the centroid document by search engine. They would be grouped into the cluster if their Kullback-Leibler divergence (KL divergence) from the centroid was below the threshold value. Any grouped documents would be removed from its document bin. At the end of each iteration, the cluster centroid document was removed from its document bin. However, when the whole clustering process ends, there would be documents left remaining unclustered. For comparison purposes, we evaluated the results with these documents grouped separately as single document clusters, and then again we evaluated the results with these documents remain "unclustered". We studied where clustering algorithm was used to improve the results of information retrieval, however, in our study, we demonstrated that retrieving candidate documents before clustering improves the performance of clustering algorithm.

To provide a solution to this problem, in this research , we proposed a authorship clustering method for Cybercrime texts, focusing on lexical features that was known as "textual fingerprints". There two phases in our algorithm, including the document retrieval phase and the document clustering phase. In information re-

trieval phase, documents that have low possibility written by the same author were filtered out before actual clustering. This increases the accuracy of our clustering algorithm as well as its efficiency. We evaluated the clustering result based on the ground truth we had. The evaluation metrics demonstrate that our proposed method has a large advantage over the baseline. Information retrieval techniques can also potentially be used in other clustering experiments.

Previous studies on authorship detection have proved the existence of “textual fingerprints” [7], we adopted this concept in our study. We considered two documents as from the same author if one of them can be retrieved by the other’s “textual fingerprints”, and at the same time, they have similarity in word probability distributions, which was measured by KL divergence. Pair-wise KL divergence calculation can be time consuming, but the workload was reduced in our study because we only calculated candidate documents that were retrieved by the fingerprint query. We used Galago Search Engine⁶ to perform document retrieval in our study.

3.3.1 Document Retrieval Phase

In the document retrieval phase, the program first preprocesses the corpus with “shingling” [11], to detect duplicate documents, and only keep one document from each duplicate set. Then, all remaining documents were put into different bins based on their word length. This was based on the characteristic of our dataset, that the same author tends to write advertisements in similar document length.

The program then calculates statistics of the corpus in this phase, including collection frequency and document frequency of each term, and term frequency in each document. This would later be the basis of forming “fingerprint query” and calculating KL divergence between any two documents.

To sum up, the preprocess work reduces the workload, and allows faster calculations in later steps.

After these preprocess, we randomly selected seed document from each document bin. Each seed document would then be deconstructed into text chunks. In our program, the size of text chunks depends on the word length of the documents. If it was longer than 200 words, it would be deconstructed into 5 text chunks. Otherwise, it would be deconstructed by each 25 words.

After text chunks were built, the program finds the term with minimal document frequency in each text chunk. Text spans around these terms would construct the “fingerprint query”. A text span was set as length of 3 words in our program.

After “fingerprint query” was built, the program retrieves candidate documents with it. The ranking function used in our study was BM25. The retrieval result

⁶<https://sourceforge.net/p/lemur/wiki/Galago/>

Table 17: Authorship Clustering Algorithm.

A. Preprocess the corpus using Shingling
B. Binning documents in corpus, initialize bin collection B
C. Calculate corpus statistics
D. Initialize cluster collection U
E. While $B \neq \emptyset$, for next bin $b_i \in B$
a. Randomly get seed document s_i
b. Generate a fingerprint query q_i and retrieve candidate documents set C_i
c. For each document d_j in C_i ,
if $dist(s_i, d_j) < \theta$:
$\forall$ cluster $u_k \in U$ with centroid document o_k ,
$o_k \neq s_i$
If $dist(s_i, d_j) < dist(o_k, d_j)$:
If cluster u_{k+1} with s_i as centroid does not exist: create $u_{k+1}, U = U \cup \{u_{k+1}\}$
$u_{k+1} = u_{k+1} \cup \{d_j\}$
Else:
$u_h = \arg \min_h (dist(o_h, d_j))$
$u_h = u_h \cup \{d_j\}$
If $d_j \in b_i : b_i = b_i - \{d_j\}$
d. Eliminate seed document s_i from bin b_i ,
$b_i = b_i - \{s_i\}$
e. If $b_i = \emptyset : B = B - \{b_i\}$
F. If $B = \emptyset$:
For all unclustered document d :
$U = U \cup d$
output U as the final clustering result

would be documents with BM25 scores higher than the threshold value α . This document retrieval step was important because it filters out a majority cybercrime documents in our dataset, and avoid calculating all pair-wise KL distance. From linguistic aspect, this step keeps those documents highly related to the fingerprint query, which can possibly indicate the same authorship, and thus increase the performance of our clustering algorithm.

3.3.2 Document Clustering Phase

The authorship clustering algorithm is shown in Table 17. In this phase, the program iterates through each candidate document and calculates distances between the seed document and each candidate document using KL divergence. In mathematical statistics, the Kullback-Leibler divergence was a measure of how one probability distribution diverges from a second, expected probability distribution [48], and a document can be considered as a unigram word probability distribution.

For any two documents d_a and d_b , assume their unigram word probability distributions were p_a and p_b , research Yang:05 proposed that the KL distance between d_a and d_b can be defined as:

$$KL(p_a||p_b) = \sum_{w_j \in d_a} p_a(w_j|d_a) \log \frac{p_a(w_j|d_a)}{p_b(w_j|d_b)} \quad (49)$$

$$p_a(w_j|d_a) = \frac{tf(w_j, d_a)}{\sum_{w_i \in d_a} tf(w_i, d_a)} \quad (50)$$

$$dist(d_a, d_b) = \min\{KL(p_a||p_b), KL(p_b||p_a)\} \quad (51)$$

If this KL distance value was smaller than the threshold value θ , we continuously calculated the KL distance between this candidate document and the centroid documents of other existed clusters. We created a new cluster if the original seed document was the nearest to the candidate document. Otherwise, we put this candidate document into the cluster whose centroid was the nearest. In the latter scenario, it means the KL distance between the candidate document and the centroid document was extremely small. Only under this circumstance, we would consider KL distance in priority to the retrieval results. Otherwise, centroid document can only be grouped with candidate documents that were retrieved by its fingerprint query. Then, we removed the candidate document from its own document bin.

At the end of document clustering phase, all documents remaining unclustered would be grouped separately as single document clusters.

3.3.3 Results and Discussion

The cybercrime dataset contained 196,078 documents. After preprocessing with Shingling, there were 86,122 unique documents left. We cleaned the documents left into TREC format. Our program then run clustering on this corpus.

In order to compare the performance of our program, three baselines and two evaluation of our best run have been computed:

1. Baseline-random randomly generates the total number of clusters and assign each document to a random cluster.
2. Baseline-singleton put each document into a single cluster.
3. Baseline- k -medoids runs k -medoids clustering based on KL distance. The number k represents the total number of authors, and it was randomly generated. Because calculating pair-wise KL distance was very time-consuming, we randomly took 1,000 documents as sample, and run k -medoids clustering on this sample corpus.
4. Baseline- n -grams uses the authorship clustering algorithm proposed by research Kocher:17. It uses a variant of the L1-norm called Canberra as distance measure function, based on the most m frequent terms or the most m frequent character n -grams of each text, to determine whether two texts were written by the same author. The code was retrieved from PAN Shared Task Series⁷ in Github. We also randomly took 1,000 documents from our corpus as sample, and evaluated the metrics based on the sample corpus.
5. IR Assisted Clustering ($\alpha = 1.0, \theta = 0.5$, one single unclustered document forms one single cluster) evaluated the result of our proposed clustering method with specific parameter values, and all unclustered documents were separately grouped as single document clusters.
6. IR Assisted Clustering ($\alpha = 1.0, \theta = 0.5$) evaluated the same result as the previous one except that all single document clusters were abandoned, and that part of ground truth would be not be used in evaluation.

The parameter α in our algorithm was the threshold value of retrieval scores. Only when a candidate document has a retrieval score higher than α , its KL distance from the seed document would be calculated, and potentially it would be grouped into a cluster. We set α to a group of different values $\{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$, and our hypotheses was that with a lower α value, more documents would be grouped into clusters, and there would be fewer documents left as single document clusters. On the other hand, a higher α value means that information retrieval techniques filter out more documents that less related to the fingerprint query. Thus, a better performs was expected with a higher α value.

The parameter θ in our algorithm was the threshold value of KL distance. A candidate document and a cluster centroid were considered belonging to the same cluster only when their KL distance was less than θ . In our experiment, θ was set

⁷<https://github.com/pan-webis-de/kocher17>

to a group of values $\{0.5, 1, 10, 100\}$. With a higher θ value, two documents were more likely to be considered belonging to the same cluster. Thus our hypotheses was that with a higher θ value, more documents would be grouped into clusters, and the “tail” of single documents would be “shorter”. At the same time, θ also represents the confidence that two documents belong to the same cluster, and how tight the cluster was. Thus, with a higher θ value, we expected a drop on the performance of our cluster. Evaluating the performance of a clustering algorithm was not as easy as only calculating the precision and recall, the metrics should reflects how similar to the ground truth set of classes this clustering algorithm separates data.

Bcubed F-measure Bcubed precision and recall was defined as the average of pairwise precision and recall of documents in the collection, and Bcubed F-measure was a harmonic mean of Bcubed precision and recall. Bcubed F-measure was known to satisfies multiple constrains, and was widely used in evaluating the performance of authorship clustering[73]. Bcubed precision and recall can be defined as

$$\text{Bcubed Precision} = \left(\sum_{i=1}^{|G|} \sum_{j=1}^{|U|} n_{ij}^2 / n_j \right) / N,$$

and

$$\text{Bcubed Recall} = \left(\sum_{i=1}^{|G|} \sum_{j=1}^{|U|} n_{ij}^2 / n_i \right) / N.$$

Then in turn, Bcubed F-measure was defined as

$$\text{Bcubed F-measure} = \frac{2 \times \text{Bcubed Precision} \times \text{Bcubed Recall}}{(\text{Bcubed Precision} + \text{Bcubed Recall})},$$

where N was the total number of documents, n_i was the size of class g_i , n_j was the size of cluster u_j , n_{ij} was the number of documents in class g_i that was put in cluster u_j , $|U|$ was the total number of clusters and $|G|$ was the total number of classes.

Homogeneity and Completeness Rosenberg’s work [63] define the homogeneity and completeness as follows:

- **Homogeneity:** each cluster contains only members of a single class.
- **Completeness:** all members of a given class were assigned to the same cluster.

Mathematically, Homogeneity and completeness scores were given as

$$h = 1 - H(G|U)/H(G),$$

		Bcubed F	Homogeneity	Completeness
1	Baseline-random	0.017	0.784	0.409
2	Baseline-singleton	0.017	0.999	0.468
3	Baseline-k-medoids	0.193	0.410	0.624
4	Baseline-n-grams	0.036	0.001	0.658
5	proposed ($\alpha = 1.0, \theta = 0.5$) ¹	0.342	0.999	0.468
6	proposed ($\alpha = 1.0, \theta = 0.5$)	0.511	0.958	0.701

¹ Each single document forms a cluster.

Table 18: Experiments for Authorship Detection

$$c = 1 - H(U|G)/H(U),$$

where $H(G|U)$ was the conditional entropy of the classes given the cluster assignments, given as

$$H(G|U) = - \sum_{g=1}^{|G|} \sum_{u=1}^{|U|} (n_{g,u}/n) \cdot \log(n_{g,u}/n_u),$$

and $H(G)$ was the entropy of the classes, given as

$$H(G) = - \sum_{g=1}^{|G|} (n_g/n) \cdot \log(n_g/n).$$

Here, n was the total number of documents, n_g and n_u were the number of documents respectively belonging to class g and cluster u , and $n_{g,u}$ was the number of documents from class g assigned to cluster u . The conditional entropy of clusters given class $H(U|G)$ and the entropy of clusters $H(U)$ were defined in a symmetric manner.

Performance of all methods, including baselines and our proposed information retrieval assisted clustering method, were compared in terms of the evaluation metrics. In Table 18, we list the performance for these method.

Among all the baselines, baseline- k -medoids achieves the highest Bcubed F-measure score. Baseline- k -medoids also has a high completeness score, even higher than our proposed method. The reason was that it only clusters all the documents from the sample corpus. It was reasonable to expect a lower completeness score when running baseline- k -medoids in the full corpus. The homogeneity score of baseline- k -medoids was the lowest, even lower than the baseline-random. It means documents from different authors were grouped into the same cluster in many

cases. That can potentially means baseline-k-medoids creates clusters less than the number of true authors, and the result from baseline-k-medoids was not very meaningful.

Meanwhile, baseline-singleton achieves almost perfect homogeneity since it provides single-item clusters exclusively. The same happens to our approach, when we assigned each single document left after clustering to a separate cluster.

At the same time, although baseline- n -grams was able to achieve a B-cubed F-score of 0.552 in the test dataset of PAN 2017 [60], it was not very competitive when used in our corpus. This again emphasizes the uniqueness of our corpus that many texts have spelling mistakes and lack sentences structures.

To more properly justify the result of our approach, we also provided the metric score after we abandoned the single document clusters. The homogeneity has a slightly drop, but both Bcubed F-measure and completeness score increase. This means that some of the single documents we left as “tail” actually belong to existing clusters. However, the existing clusters was highly possible to only have documents from single author, according to the homogeneity score.

3.4 Innovative Search User Interface - Virtual Reality

The state-of-the-art search engine interface was boring. The static design of a square query box and ten blue links has been there for years. It limits the number of ways that a user can interact with a search engine. It also limits the types of input signals that a search algorithm can collect from the user. In this research, we introduced a Virtual Reality search engine interface. Virtual reality (VR) has been explored in the game industry and multimedia community for a while. In the working environment or the office settings, VR’s potential was understudied. As a first step to enable a VR-supported working environment, we presented a search engine supported by virtual reality. As far as we know, it was probably the world’s first VR-enhanced search engine. Users can dive into an immersive VR working environment, in which they can read, search and interact with a search engine without aids of keyboard or mouse.

Virtual reality had been a popular topic in our society for a while. It offers a new and exciting way of presenting digital content. Once a user was wearing a virtual reality device, a realistic experience was simulated around the user. VR’s potential in technology and in marketing was far under-explored. Thanks to the more affordable price of VR devices, it was very likely that we would witness a popularization of VR technologies [85]. As the VR devices were more and more available to the public, research on VR has shifted from primarily focusing on making VR possible (hardware) to applying the technique to various domains – a shift from hardware to applications [67]. Current VR applications span across



Figure 27: Curved Text Display Interface

the game industry [51, 89], education industry [5, 43, 58, 65, 71], multimedia industry [9, 17], and medical industry [21, 59, 64]. Unfortunately, as far as we know, little research has been done using VR in the office settings. In the 1980s, most people use papers in their offices. In the 1990s, most people use computers in their offices. We asked ourselves: was there a possibility that after a few decades, people in the office would only need to use their hands to grab whatever they need in a virtual reality world, just like what was shown in the “minority report” movie [72]. In today’s office, people spend a deal of of time using search engines. As a first step to enable a VR-supported working environment, we developed a novel VR interface for search engine.

Regardless of VR, search engine interface has not changed much in the past decade. The interface design of a square query box and ten blue links has been there forever. We were not in a position to judge its effectiveness nor efficiency. However, we believed it reduced the possibilities of new forms of user signals to improve search engine effectiveness. In this research, we showed our exploration in developing a virtual reality interface for search engine. In Hearst’s book [27], she brought up a search engine interface design principle: “Search user interfaces must be understandable by and appealing to a wide variety of people of all ages, cultures and backgrounds, and for an enormous variety of information needs.” We followed this principle in our design and deployment. Our VR-enabled search engine adopted VR, which to most users would be a drastic change of user experience. To encourage users to overcome the fear to new things, a new interface



Figure 28: Virtual Keyboard

must be simple and clear [6]. We also followed this principle in our design and deployment.

3.4.1 Curved Text Display for Reading and Interacting with Documents

Virtual reality was well-known to render impressive 3D landscapes, however text was generally difficult to be read in Virtual Reality settings. It has been challenging for VR developers to render text. The main reason was that displaying text requires to render many details that were small in size. The resolution of VR devices in the market was too low to support such high quality display for text. Besides that, text was 2D. Therefore it gets blurry when a letter was displayed from a far distance to the center of the view.

In this research, we found ways to achieve better quality of text display with the current VR technologies. Our solution was the following. We tested text and images on several different types of surfaces in VR. The surfaces include flat plane, sphere, and cylinder. And we recruited several users to test the visibility of both text and images. We found that the cylinder surface generate the best user experience. On a flat plane, the text and images were distorted on the far end of the view, which makes them very difficult to be recognized visually. Generally, users can see clearer text and images on a curved surface than on a straight surface. For the area that was just in front of a user, the sphere surface and the cylinder



Figure 29: Grab a Keyword

surface look similar . However, for the area that was top of a user's view, the text and images become very small on a sphere surface. We therefore selected the cylinder surface to display a document for our search engine. Figure 28 shows the curved document interface.

There was no list representation in our display. Although the top ranked document was displayed right in front of a user, the user would need to use hand gestures to browse to the next document. We thought eliminating the top-down ranked list would remove user' bias about ranking positions. Such biases include that users tend to click on top ranked documents more regardless its true relevance. Our new search result display might open up opportunities for fairer search results evaluation.

3.4.2 Keyword Queries

Traditional computer accessories, such as mice and keyboards were difficult to manage in VR settings. It was because users would usually walk around in her VR world. Our basic guideline was to minimize the number of accessories. Mouse and keyboard were completely abandoned in our user interface design. Instead, we supported users to use hand gestures such as tap to substitute a mouse click. Moreover, to get rid of keyboards, we invented a virtual keyboard for our VR search engine. A user can still type in query keywords on the virtual keyboard. When a user tabs the keyboard button, the virtual keyboard pops up in front of

the user. The virtual keyboard then occupies almost half of front view, which was big and easy for the user to tap letters to enter queries. The tapped letters would show in front of the user, floating on top of the virtual keyboard. The reason that we displayed the typed keywords overlapping in front of the virtual keyboard instead of being above or below it was that, the virtual keyboard was so big that a user would move her head drastically to try to view the entire keyboard if the typed keywords were shown right in front of her. Such big head movements should be minimized due to its impact to the head device. During our test phase, we find that users almost constantly move their hands across different keys. For example, when a user moves his finger from “A” to “F”, he would be very easily to touch “S” and “D” which were located between “A” and “F”. To eliminate the mistype, we implemented a cool-down timer for every object in the virtual keyboard. Each time when a key was touched, the timer starts to count down. The keyboard was not activate to any user action until the count down was finished.

To minimize user efforts, we enabled the user to grab a keyword to search from the document that he was currently reading (Figure 29). When the user finds some interesting keywords in a document, she can grab the keywords from the document and put it into a query box. The process was , first the user moves his hand close to the document. The word closest to the hand would be highlighted. Next, the user holds her hand into a fist, a colored (set to red) duplication of the highlighted word would be created and follows the hand’s movement. Then, the user keeps holding his fist and moves the hand towards the query box. The query box would be highlighted when a hand was approaching. In the end, the user releases his fist to drop the red duplication, i.e. the highlighted keyword from the document, into the query box. Here our query box was literally a box of 3D dimension, which was used to hold query words.

3.4.3 Structural Queries

Non-structural keyword queries do not specify the relationships among the keywords. Figure 30 shows how we handled an AND query. Structural queries not only give the keywords, but also specify the relationships. Common search engine operators include “And” and “Or”. Many users like to use structural queries to stay control in combining or intersecting search results. To meet this requirement, we provided novel VR functions for users to manipulate the keywords. In particular, users can specify “AND”, “OR” relationships among keywords by using hand gestures.

How does a user specify the “OR” relationship between two keywords? For example, if a user puts her hand next to a keyword “counterfeit“ in the text, and the keyword turns red. Next, the user closes her hand making a fist. It was how



Figure 30: AND query operator

the user “grab” the keyword into her hand. Then, the user moves the keyword in her hand to the query box. All individual keywords in the query box were considered using the “OR” operator by default. If a query has two words, the user can wave her hand from top-right to bottom-left to split the query into two words. And these two were now in an “OR” relationship.

How does a user specify the “AND” relationship between two keywords? If in the query box there were two keywords (for example “counterfeit” and “electronic”), the user moves both of her hands next to the two keywords, one hand for each word. The two words would turn red. Then, the user closes her hands into fists to lock the two keywords. A keyword’s background turns green when it was locked. Last, the user claps her two fists to combine the two keywords into an “AND” relationship. Keywords were connected by the AND operators were put into rectangles. These AND rectangles can be used as components for other “AND” or “OR” operators. We could thus form conjunctive normal form (CNF) clauses as structural search queries.

3.4.4 Increasing Gesture Detection Accuracy

The user can interact with the 3D-displayed text with hands. We used a device called “Leap Motion” to capture hand movements. It contains a camera shooting at the hands and a processor analyzing the captured images. The data generated by Leap Motion were positions and rotations of hands’ joints and bones. Our

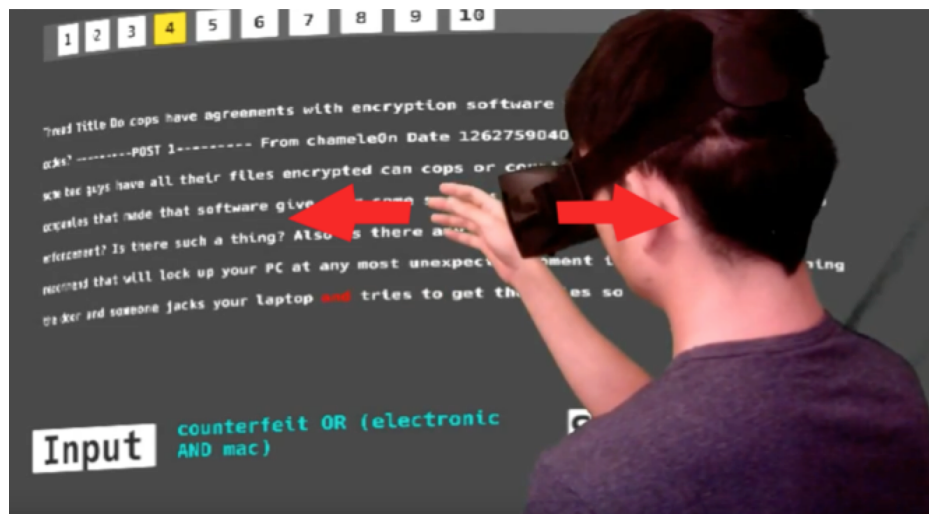


Figure 31: Swipe

search engine uses them to detect gestures. The detection accuracy can be improved by optimizing our gesture detection algorithm, and/or by optimizing Leap Motion’s image processing algorithm. Leap Motion upgrades its product fast and constantly improves its precision. However, there was definitely room to improve. Nonetheless, with the current detection accuracy provided by Leap motion, we believed our tool was acceptable to regular search engine users. Next, we described a few gesture detection challenges that we have tackled.

The first gesture we recognized was swipe (Figure 31). A user swipes their hands to navigate through a list of documents. We should be aware that a user would move her hand back and forth regularly. Therefore the challenge here was to distinguish the document navigation swipe gestures from regular hand movements. We employed a simple decision-tree-like thresholding method to classify them. We used a duration threshold, a distance threshold and an angle threshold. A user was making a swipe gesture only when her hand moves fast enough, far enough, and the hand was perpendicular to the horizon. We used these three thresholds to filter out irrelevant movements.

The second gesture we recognized was tap. A user taps regularly when she uses our VR keyboard. Several gesture detection errors could easily frustrate the user: 1) The user wants to tap the letter key “S”, but she touches the left edge of key “S”. Therefore she also taps key “A” at the same time. 2) The user wants to tap key “S”. However on her way to “S”, her hand accidentally touches “D” first.



Figure 32: Fist Detection

3) She only wants to tap key “S” once. However because of accidentally shaking of camera or hands, she taps it twice and “S” were inputted twice. We proposed two mechanisms to solve the problems. First, a fingertip has six probes around it. Each probe can be used to detect a collision with an object. Only if five or more probes were collided with the same object, we activated a tap event. In this way, we could eliminate the chance that a user taps multiple objects at the same time, thus reduce the false positives. Second, we set a cool-down counter on objects. When a user taps an object, the object activates a tap event and hibernates for a while (for example 0.2 seconds). In this way, we could reduce the chance of redundant taps, too.

The third kind of gestures we recognized was making a fist and then opening the hand. We tried two approaches to detect this. The first was that we measured the distance between hand joints. If the joints were close enough to each other, then the hand was making a fist. The second approach was to measure the angle between the bones in fingers. If the fingers were bending inwards, the hand was making a fist. Our result shows that the second approach works better. Another challenge was that the device can’t perfectly model the position and rotation of the joints and bones. Sometimes the hand jerks back and forth. To eliminate the small movements, we used timeout thresholding. We recognized a posture only if it lasts over a certain amount of time (set to 0.2 seconds). Figure 32 shows the fist detection.

3.4.5 Results and Discussion

Most users might not be ready to use VR. When people were still in the process of acquiring new user experience, sometimes they feel exciting, other times they feel challenged and confused. To minimize negative effects, we devoted a great deal of effort to develop features aiding users while they were using this novel VR-enhanced search engine. Our goal was to increase the system transparency to the users and make sure they know where they are. Those features include animation and highlighting.

Animation influences the user's eyes and it can control where they should focus. A good animation can attract the user's attention to a right place. For example, an animation-free method of grabbing a word from a document would be that, the user puts her hand in front of the word and closes her hand into a fist, then the word magically appear in the query box. This method would leave the user to wonder "what just happened when I closed my hand?", "why did the word suddenly appear in the query box?", etc. By adding animations, like making the grabbed word following the hand's movement and allowing the grabbed word to be dropped slowly into a box, instead of suddenly appears in the query box, we could greatly reduce the amount of user confusion by slowing the process and allowing the user to participate in the process. The same idea applies to the highlighting function. We added many highlighting features to guide the user through our workflow. For example, when hands were close to the query box, the query box turns into green to indicate a short distance.

4 Tools and Deliverables

We developed and deployed quite a few software tools during the three-year period. Most of them can be found at <http://infosense.cs.georgetown.edu/>. Below was a list of each individual tool's information.

- **Dumpling.** It was called Direwolf originally and changed to dumpling. It has side-by-side search functionality to compare dynamic search vs. google-like non-dynamic search. Figure 33 shows the interface of Dumpling.
- **Minerva** It was a document collection exploration and search and tagging tool to allow a user to discover and save and extract information. Figure 34 shows the interface of the Minerva tool.
- **Minerva II** was an entity detection and linking tool. Part of it was **Minerva Visual**, which displays complex network structures in a neat way.

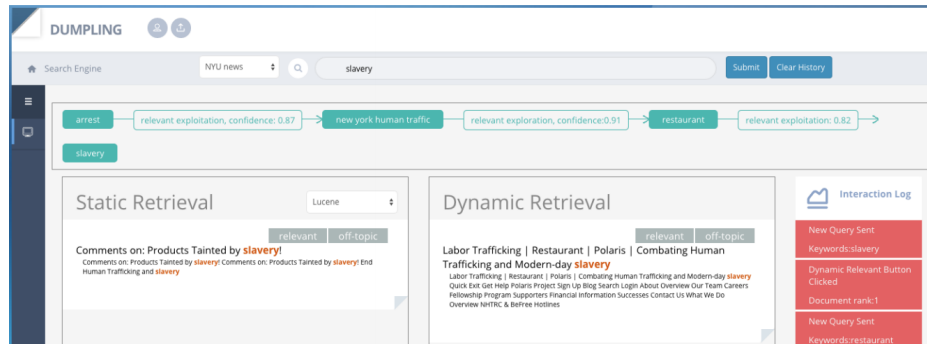


Figure 33: Dumpling - A Dynamic Search Engine

Figures 35 and 36 show the interfaces of Minerva II and Minerva Visual respectively.

- **TREC Dynamic Domain Annotation, Administration, and Simulated User Package.** These were the several tools that we built for TREC Dynamic Domain Track. They were released to the public. Figures 37 and 38 show the tool interfaces of the annotation tool and the jib respectively.
- **Counter-Human-Trafficking Faceted Search Engine.** The faceted search engine and QA system that we built for counter human trafficking. Figure 39 shows the interface of the Counter-Human Trafficking Faceted Search Engine.
- **VR Search** The Virtual Reality tool that we built for dynamic search user interface. Figure 40 shows the VR user interface.

5 Conclusions

In this research, we firstly presented a novel POMDP framework to model dynamic search for domain-specific search. In the framework, the user and the search engine were modeled as two agents who cooperate to achieve a common goal which was fulfilling the user's information need with minimized effort. The framework was built upon the Partially Observable Markov Decision Process model. By carefully designing states, actions, and observations, the new framework can efficiently optimize search engines' performance. We designed states as user mental statuses during search. In detail, we modeled two different dimensions: 1)

Pattern Mining - Minerva I

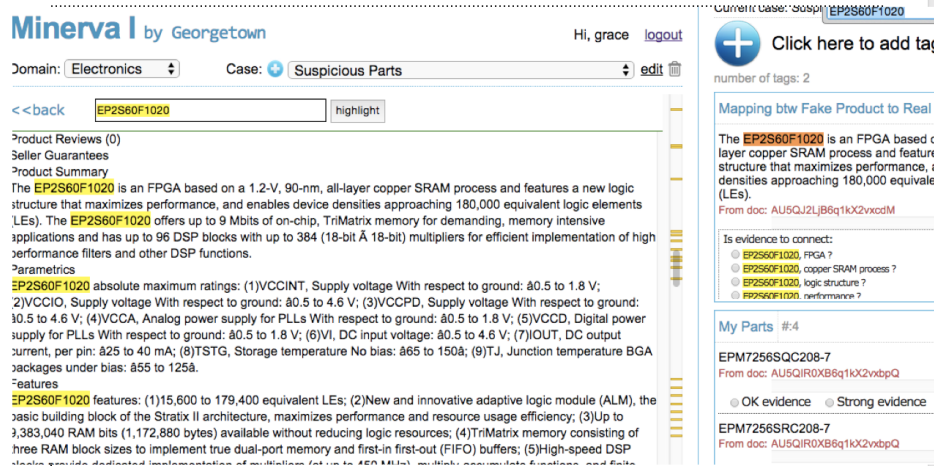


Figure 34: Mineva

Evidence Box - Minerva II

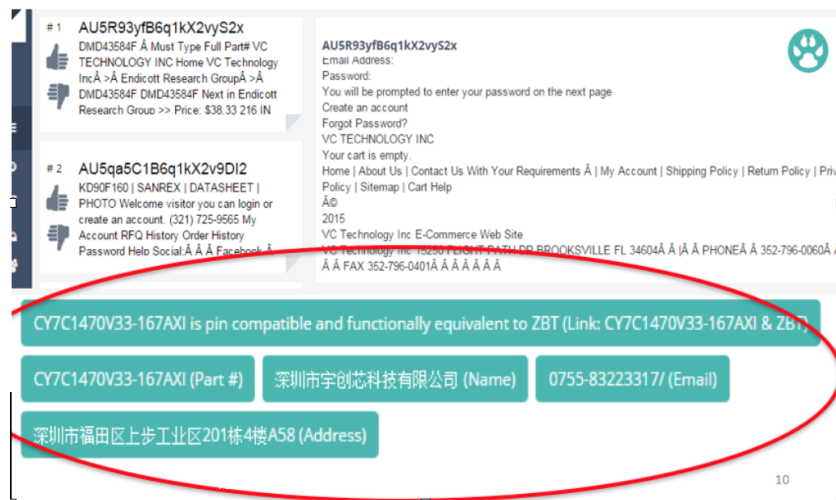


Figure 35: Mineva II

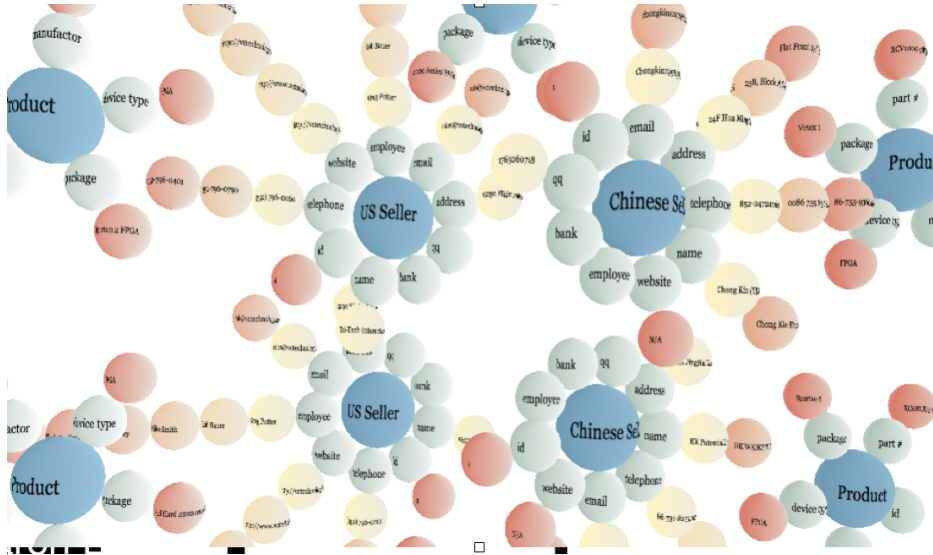


Figure 36: Mineva Visual

TREC Dynamic Domain Annotation Tool

TREC-DD Annotation by Georgetown

Hi, grace [logout](#)

Domain: Topic: [edit](#)

[Search](#) [recommend](#) **Search result**

Searched for: military response Results 1 - 50 of 1000 matches

1. ebola-8252ab7ca664523c2fa855fb42d066ca3e4bd062ab42bc8445fcaab449d43b
Inter Press Service » U.S. Military Joins Ebola Response in West Africa <http://www.ipnews.net/Turning-the-World-Downside-UpTue,17Mar201520:39:38+0000en-UShourly1http://wordpress.org/?v=4.1.1> U.S. Military Joins Ebola Response in West Africa http://www.ipnews.net/2014/09/u-s-military-joins-ebola-response-in-west-africa/?utm_source=rss&utm_medium=rss&utm_campaign=u-s-military-joins-... (Score: -5.28844)
2. ebola-b51d48f45822c7d8f4a7105b5c959038d9cbab333563584d515cab901b925c
Inter Press Service » U.S. Military Joins Ebola Response in West Africa <http://www.ipnews.net/Turning-the-World-Downside-UpSat,10Jan201511:56:54+0000en-UShourly1http://wordpress.org/?v=4.1.1> U.S. Military Joins Ebola Response in West Africa http://www.ipnews.net/2014/09/u-s-military-joins-ebola-response-in-west-africa/?utm_source=rss&utm_medium=rss&utm_campaign=u-s-military-joins-... (Score: -5.28844)
3. ebola-c832abac905448-7fac18749ab77b0ff67411ed4d0a6c3c1caebd096055
...Civil Liberties, HIPAA, Privacy Research Health IT Research and Development Technology Information Technology Supporting the **Military** Health System Recent News Articles 1/8/2015 Anesthesiologists keep soldiers safe in Afghanistan 1/8/2015 DoD Releases Breakdown of Ebola Response Effort 1/8/2015 E-Prescribing – Coming Soon to a **Military** Pharmacy Near You! Training opportunities in the **Military** Health System Training for MHS Staff Addressing the Psychological Health of Warriors and their Families... (Score: -5.3626)
4. ebola-b972b97dbfab76105e0c35ac2abc108f811deb1295521874338d6f132d109a
...Civil Liberties, HIPAA, Privacy Research Health IT Research and Development Technology Information Technology

[viewed doc](#) [completed doc](#) [discarded doc](#)

Review result

Subtopic drop box

number of subtopics: 2

temp #2

From doc: ebola-c092b6c965a20d1164b8a1c5caf1e478f
☐ marginally relevant ☒ relevant ☐ highly relevant
 result

17 Mar 2015 20:39:38 +0000en-UShourly1http://www.ipnews.net/2014/09/u-s-military-joins-ebola-response-in-west-africa/?utm_source=rss&utm_medium=rss&utm_campaign=...
 From doc: ebola-8252ab7ca664523c2fa855fb42d066ca3e4bd062ab42bc8445fcaab449d43b
☐ marginally relevant ☒ relevant ☐ highly relevant
 result

Reuter #2

Ebola CausesTue, 30 Dec 2014 18:40:24 +0000en-UShourly1http://wordpress.org/?v=4.1.1
<http://www.ebolacauses.com/us-army-completes-treatment-unit-2/> <http://www.ebolacauses.com/us-army-completes-treatment-unit-2/#comments> Fri, 19 Dec 2014 06:33 +0000http://www.ebolacauses.com/?p=1385The U.S. Army engineers completes Ebola treatment center in Liberia. Subscribe
 From doc: ebola-c092b6c965a20d1164b8a1c5caf1e478f
☐ marginally relevant ☒ relevant ☐ highly relevant
 result

[View annotations](#) [Write](#)

Figure 37: TREC Dynamic Domain Track Annotation Tool

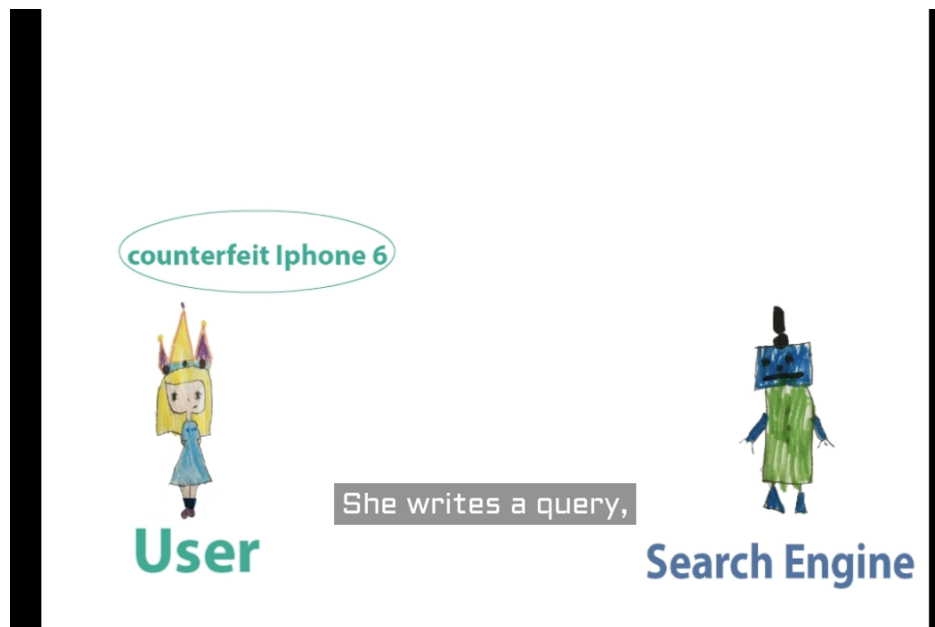


Figure 38: TREC Dynamic Domain Track Simulated User (Jig)

whether the user finds relevant content for previous query; 2) whether the user wants to explore or exploit for the current query. The framework chooses the optimal retrieval algorithm for the search engine by considering the user's mental status during search. The results show that our framework was highly effective for dynamic search. We hope the work would inspire future adventures in the fields of POMDP in IR and game theory in IR.

Second, in order to improve the efficiency of dynamic search, we introduced another reinforcement learning algorithm – direct policy learning. It was also based on the POMDP framework. It produces an optimal search strategy directly from the observations collected during search process, which reduces the complexity of the original model. In addition, it was able to flexibly incorporate a wide range of features derived from the rich interactions between the user and the search engine in dynamic search. Our experimental results show that the direct policy learning algorithm was a statistically significant improvement over the best TREC runs in 2012 and 2014 and in DARPA Memex evaluation for point-fact questions.

Third, we also implemented dynamic search functions to support the annotation and organization task for the TREC Dynamic Domain Track

Search Interface



1. Structured and unstructured query; 2. Subtopic drag and drop search; 3. Review search result

Figure 39: Counter-Human-Trafficking Search Engine



Figure 40: Virtual Reality Search Engine User Interface

from 2015 to 2017. This was our first trial of implementing a dynamic search system to tackle a real dynamic search problem. In the end, our annotation tool provides three ad-hoc search functions (Lemur, Solr and Terrier), one relevance feedback search function and two dynamic search functions. We compared the performance of these functions using various IR metrics, including Recall, Precision@10, MAP, nDCG@10 and α -nDCG. The evaluation results show that dynamic search outperforms the ad-hoc search functions over all evaluation metrics, and it also outperforms relevance feedback over Recall, Precision@10 and MAP. The experimental results successfully show that dynamic search was effective in real life search scenarios.

Fourth, we implemented our dynamic search engines and apply them for question answering. We won top positions in the Memex point-fact answer evaluation.

Fifth, we studied the authorship detection problem in the cyber-crime datasets provided by Memex. Our algorithm uses fingerprints to detect writing styles and works quite effective in grouping documents written by the same criminal.

Lastly, we also invented a novel search engine user interface by creating the search UI in the virtual reality settings.

The primary goal of this research was to provide solutions for modeling dynamic search effectively and efficiently. The significance of these solutions proposed in this research , and its potential impact were discussed in this section.

Before this work, researchers mainly identified the user as the only agent who drove the search process in dynamic search. The user decided what to search and what document fits his/her information need. The search engine just passively respond to user commands. Our first significant contribution of this work is that it realized that there were actually two agents in the process, the user agent and the search engine agent. Sometimes the user agent had a clear goal of what s/he was looking for, but it was not always the case. Even when the user had a clear goal, search engine still have to make a lot of decisions during search. For example, when the user repeated a search query, the search engine could decide to return the same materials that the user had already showed interests or to provide some novel content. Acknowledging that the user and the search engine worked together to decide the search path, we modeled dynamic search as a dual-agent stochastic game for both agents to maximize their long term goal – fulfilling the information need. The framework enableed the optimization of the search strategies for both agents.

Researchers noticed the rich user interactions in dynamic search for a long time. However, user intents change rapidly in dynamic search. Without correctly modeling the dynamics during search, rich user interactions could not be effec-

tively utilized. For example, without knowing whether the user still interested in the same topic, we could not decide whether we should use the relevance signals regarding such topic to reinforce retrieval results. The dynamic search frameworks proposed in this work were based on RL frameworks, which well models the rich interactions, and temporal dependency of search states in dynamic search. The second major contribution was that this work proposes retrieval models that significantly improved the search accuracy of state-of-the-art dynamic search algorithms.

Moreover, we implemented the proposed RL frameworks in a dynamic search tool to support the annotation task for the TREC Dynamic Domain Tracks. This was a dynamic search scenario in real life setting. Real users utilize our tool to discover interesting topics and their relevance content in the corpus. The annotation results demonstrate that the retrieval models proposed in this research were not only effective and efficient at theoretical levels in laboratory settings. They were ready to be used in real world dynamic search scenarios.

Our research also addressed the efficiency problem when applying RL frameworks to model dynamic search. Search satisfaction was highly related to retrieval latency [4], however, most RL frameworks were computational expensive. Without solving the problem of efficiency, it was unrealistic to apply RL frameworks in real life dynamic search scenarios. In this research, we limited the number of states and actions in our POMDP frameworks to reduce the demands of computation. We further improved the efficiency by eliminating the necessary of calculating states in RL frameworks and directly learning optimal policies from observations.

Other than the main contribution area – reinforcement learning for dynamic search – this research made achievements across multiple related disciplines: Machine Learning, Information Retrieval, Text Mining, Natural Language Processing/Understanding, and Human-Computer Interaction. We explored many research topics, including Reinforcement Learning, Dynamic Search, Faceted Search, Question Answering, Entity Linking, Authorship Detection, Visualization, and Search Engine Evaluation.

What we had learned from the three-year work was the following: 1) we must recognize the importance of using good methods to incorporate domain knowledge in a search system; 2) it was challenging yet necessary to merge techniques from machine learning, information retrieval, database and data mining. Classical free text retrieval needs to satisfy users' expectation of classical database-like queries, result presentations, visualizations. 3) Search was a process. We proposed many algorithms for dynamic search in this project, however new tools and UI designs were still awaiting the researchers to pursue. Nonetheless, In this research, we proudly invented and improved the state-of-the-art dynamic search algorithms and systems for domain-specific search.

6 Appendix - Acronyms

1. 3D: three-dimensional
2. ACM: Association of Computing Machinery
3. AI: Artificial Intelligence
4. AND: and
5. BM25: Okapi BM25, a retrieval algorithm
6. CHI: Computer Human Interaction
7. CHT: Counter Human Trafficking
8. CIKM: Conference on Information and Knowledge Management
9. CLK: click
10. ClueWeb: name of a web document collection
11. ClueWeb CatB: ClueWeb Category B
12. CT: Cube Test
13. DARPA: Defense Advanced Research Projects Agency
14. DB: Database
15. DCG: Discounted Cumulative Gain
16. DD: Dynamic Domain
17. DM: Data Mining
18. DPL: Direct Policy Learning
19. DUP: Duplicate
20. ECIR: European Chapter of Information Retrieval
21. EM: Expectation Maximization
22. EU: Expected Utility
23. F: F-measure

- 24. ICTIR: International Conference on Theory of Information Retrieval
- 25. idf: Inverse document frequency
- 26. IR: Information Retrieval
- 27. IRJ: Information Retrieval Journal
- 28. LDA: Latent Dirichlet Allocation
- 29. Lemur: name of a search engine software package
- 30. LinUCB: Lin Upper Confidence Bound
- 31. LM: Language Modeling
- 32. MAP: Mean Average Precision
- 33. MDP: Markov Decision Process
- 34. ML: Machine Learning
- 35. MLE: Maximum Likelihood Estimation
- 36. nCT: normalized Cube Test
- 37. nDCG: Normalized Discounted Cumulative Gain
- 38. nERR: Normalized Expected Reciprocal Rank
- 39. NIST: National Institute of Standards and Technology
- 40. NLP: Natural Language Processing
- 41. NP-Hard: Non-deterministic Polynomial-time Hardness
- 42. OR: or
- 43. P: precision
- 44. PHD: Doctor of Philosophy
- 45. POMDP: Partially Observable Markov Decision Process
- 46. QCM: the Query Change Model
- 47. R: recall
- 48. RL: Reinforcement Learning

- 49. s.t.: Such That
- 50. SAT: Satisfactory
- 51. SAT Click: Satisfactory Click
- 52. sDCG: Session Discounted Cumulative Gain
- 53. SIGIR: Special Interest Group on Information Retrieval
- 54. tf: term frequency
- 55. TOIS: Transaction on Information Systems
- 56. TREC: Text REtrieval Conference
- 57. TREC DD: Text REtrieval Conference Dynamic Domain
- 58. UCB: Upper Confidence Bound
- 59. UI: User Interface
- 60. UK: the United Kingdom
- 61. US: the United States
- 62. VR: Virtual Reality
- 63. WSDM: Web Search and Data Mining

References

- [1] A. Al-Maskari and M. Sanderson. A review of factors influencing user satisfaction in information retrieval. *Journal of the American Society for Information Science and Technology*, 61(5):859–868, 2010.
- [2] A. Albahem, L. Cavedon, D. Spina, and F. Scholer. Rmit @ trec 2016 dynamic domain track: Exploiting passage representation for retrieval and relevance feedback. In *TREC '16*, 2016.
- [3] K. Aldebei, H. Farhood, W. Jia, P. Nanda, and X. He. Sequential and unsupervised document authorial clustering based on hidden markov model. In *24th IEEE Trustcom/BigDataSE/ICSS*, pages 379–385, 2017.
- [4] I. Arapakis, X. Bai, and B. B. Cambazoglu. Impact of response latency on user behavior in web search. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, pages 103–112, New York, NY, USA, 2014. ACM.
- [5] E. Arroyo and J. L. L. Arcos. Srv: a virtual reality application to electrical substations operation training. In *Multimedia Computing and Systems, 1999. IEEE International Conference on*, volume 1, pages 835–839. IEEE, 1999.
- [6] A. Aula and M. Käki. Less is more in web search interfaces for older adults. *First Monday*, 10(7), 2005.
- [7] H. Baayen, H. van Halteren, A. Neijt, and F. Tweedie. An experiment in authorship attribution. In *6th JADT*, pages 29–37, 2002.
- [8] L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 37(6), 1966.
- [9] D. R. Begault and L. J. Trejo. 3-d sound for virtual reality and multimedia. 2000.
- [10] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
- [11] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. *Computer Networks and ISDN Systems*, 29(8-13):1157–1166, 1997.
- [12] G. Cao, J.-Y. Nie, J. Gao, and S. Robertson. Selecting good expansion terms for pseudo-relevance feedback. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 243–250, New York, NY, USA, 2008. ACM.

- [13] B. Carterette. An analysis of np-completeness in novelty and diversity ranking. In *ICTIR'2009*, pages 200–211. Springer, 2009.
- [14] B. Carterette, E. Kanoulas, and E. Yilmaz. Simulating simple user behavior for system effectiveness evaluation. In *CIKM '11*.
- [15] M.-A. Cartright, R. W. White, and E. Horvitz. Intentions and attention in exploratory health search. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 65–74, New York, NY, USA, 2011. ACM.
- [16] L. B. Chilton and J. Teevan. Addressing people’s information needs directly in a web search result page. In *WWW '11*.
- [17] C.-C. P. Chu, T. H. Dani, and R. Gadh. Multi-sensory user interface for a virtual-reality-based computeraided design system. *Computer-Aided Design*, 29(10):709–725, 1997.
- [18] C. L. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon. Novelty and diversity in information retrieval evaluation. In *SIGIR'2008*, pages 659–666. ACM, 2008.
- [19] A. Daks and A. Clark. Unsupervised authorial clustering based on syntactic structure. In *Proceedings of the 54th ACL Student Research Workshop*, pages 114–118. Association for Computational Linguistics, 2016.
- [20] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *J. Roy. Statist. Soc. Ser. B*, 39(1):1–38, 1977.
- [21] J. Difede, J. Cukor, I. Patt, C. Giosan, and H. Hoffman. The application of virtual reality to the treatment of ptsd following the wtc attack. *Annals of the New York Academy of Sciences*, 1071(1):500–501, 2006.
- [22] Y. García-Mondeja, D. Castro-Castro, V. Lavielle-Castro, and R. Muñoz. Discovering Author Groups using a B-compact graph-based Clustering—Notebook for PAN at CLEF 2017. In *CLEF 2017 Evaluation Labs and Workshop – Working Notes Papers, 11-14 September, Dublin, Ireland*. CEUR-WS.org, 2017.
- [23] H. Gómez-Adorno, Y. Alemán, D. Vilariño Ayala, M. Sanchez-Perez, D. Pinto, and G. Sidorov. Author Clustering using Hierarchical Clustering Analysis—Notebook for PAN at CLEF 2017. In *CLEF 2017 Evaluation Labs and Workshop – Working Notes Papers, 11-14 September, Dublin, Ireland*. CEUR-WS.org, 2017.
- [24] D. Guan, H. Yang, and N. Goharian. Effective structured query formulation

- for session search. In *Proceedings of the 21st Text REtrieval Conference, TREC'12*, 2012.
- [25] D. Guan, S. Zhang, and H. Yang. Utilizing query change for session search. In *SIGIR '13*.
 - [26] Q. Guo and E. Agichtein. Ready to buy or just browsing?: detecting web searcher goals from interaction data. In *SIGIR '10*, pages 130–137, 2010.
 - [27] M. Hearst. *Search user interfaces*. Cambridge University Press, 2009.
 - [28] K. Hofmann, S. Whiteson, and M. de Rijke. Balancing exploration and exploitation in learning to rank online. In *ECIR'11*, pages 251–263, 2011.
 - [29] S. B. Huffman and M. Hochster. How well does result relevance predict session satisfaction? In *SIGIR'2007*, pages 567–574. ACM, 2007.
 - [30] J. L. S. Z. Hui Yang, Xuchu Dong. Session search modeling by partially observable markov decision process. *Information Retrieval Journal (IRJ)*, 20, 2017.
 - [31] S. Z. Hui Yang, Dongyi Guan. The query change model: Modeling session search as a markov decision process. *ACM Transactions on Information Systems*, 33(4), 2015.
 - [32] F. Iqbal, H. Binsalleeh, B. C. Fung, and M. Debbabi. Mining writeprints from anonymous e-mails for forensic investigation. *digital investigation*, 7(1):56–64, 2010.
 - [33] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.
 - [34] K. Järvelin, S. L. Price, L. M. Delcambre, and M. L. Nielsen. Discounted cumulated gain based evaluation of multiple-query ir sessions. In *ECIR'2008*, pages 4–15. Springer, 2008.
 - [35] J. Jiang, D. He, and S. Han. Pitt at trec 2012 session track. In *TREC '12*, 2012.
 - [36] X. Jin, M. Sloan, and J. Wang. Interactive exploratory search for multi page search results. In *Proceedings of the 22Nd International Conference on World Wide Web, WWW '13*, pages 655–666, New York, NY, USA, 2013. ACM.
 - [37] T. Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning, ICML '97*, pages 143–151, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

- [38] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1):99–134, 1998.
- [39] E. Kanoulas, B. Carterette, M. Hall, P. Clough, and M. Sanderson. Overview of the trec 2011 session track. In *TREC'11*.
- [40] E. Kanoulas, B. Carterette, M. Hall, P. Clough, and M. Sanderson. Overview of the trec 2012 session track. In *TREC'12*.
- [41] E. Kanoulas, B. Carterette, M. Hall, P. Clough, and M. Sanderson. Overview of the trec 2013 session track. In *TREC'13*.
- [42] E. Kanoulas, B. Carterette, M. Hall, P. Clough, and M. Sanderson. Overview of the trec 2014 session track. In *TREC'14*.
- [43] H. Kaufmann, D. Schmalstieg, and M. Wagner. Construct3d: a virtual reality application for mathematics and geometry education. *Education and information technologies*, 5(4):263–276, 2000.
- [44] Y. Kim, A. Hassan, R. W. White, and I. Zitouni. Modeling dwell time to predict click-level satisfaction. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining, WSDM '14*, 2014.
- [45] M. Kocher and J. Savoy. UniNE at CLEF 2017: Author Clustering—Notebook for PAN at CLEF 2017. In *CLEF 2017 Evaluation Labs and Workshop – Working Notes Papers, 11-14 September, Dublin, Ireland*. CEUR-WS.org, 2017.
- [46] T. Kokubu, T. Sakai, Y. Saito, H. Tsutsui, T. Manabe, M. Koyama, and H. Fujii. The relationship between answer ranking and user satisfaction in a question answering system. In *NTCIR'2005*, 2005.
- [47] A. Kotov, P. N. Bennett, R. W. White, S. T. Dumais, and J. Teevan. Modeling and analysis of cross-session search tasks. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '11*, pages 5–14, New York, NY, USA, 2011. ACM.
- [48] S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.
- [49] R. Layton, P. Watters, and R. Dazeley. Automated unsupervised authorship analysis using evidence accumulation clustering. *Natural Language Engineering*, 19(1):95–120, 2013.
- [50] Lemur Search Engine. <http://www.lemurproject.org/>.
- [51] D. Levac, M. R. Pierrynowski, M. Canestraro, L. Gurr, L. Leonard, and

- C. Neeley. Exploring childrens movement characteristics during virtual reality video game play. *Human movement science*, 29(6):1023–1038, 2010.
- [52] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. *WWW '10*, 2010.
 - [53] J. Liu and N. J. Belkin. Personalizing information retrieval for multi-session tasks: The roles of task stage and task type. In *SIGIR '10*, 2010.
 - [54] J. Luo, C. Wing, H. Yang, and M. Hearst. The water filling model and the cube test: multi-dimensional evaluation for professional search. In *CIKM'2013*, pages 709–714. ACM, 2013.
 - [55] J. Luo, S. Zhang, and H. Yang. Win-win search: Dual-agent stochastic game in session search. In *SIGIR '14*.
 - [56] H. Y. Marc Sloan and J. Wang. A term-based methodology for query reformulation understanding. *Information Retrieval Journal (IRJ)*, 18:145–165, 2015.
 - [57] J. R. Norris. *Markov Chains*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1997.
 - [58] K.-M. Park, J. Ku, S.-H. Choi, H.-J. Jang, J.-Y. Park, S. I. Kim, and J.-J. Kim. A virtual reality application in role-plays of social skills training for schizophrenia: a randomized, controlled trial. *Psychiatry research*, 189(2):166–172, 2011.
 - [59] S. Parsons and S. Cobb. State-of-the-art of virtual reality technologies for children on the autism spectrum. *European Journal of Special Needs Education*, 26(3):355–366, 2011.
 - [60] M. Potthast, F. Rangel, M. Tschuggnall, E. Stamatatos, P. Rosso, and B. Stein. Overview of PAN'17: Author Identification, Author Profiling, and Author Obfuscation. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction. 7th International Conference of the CLEF Initiative (CLEF 17)*. Springer, 2017.
 - [61] K. Raman, P. N. Bennett, and K. Collins-Thompson. Toward whole-session relevance: Exploring intrinsic diversity in web search. In *SIGIR '13*, 2013.
 - [62] S. Robertson and H. Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, Apr. 2009.
 - [63] A. Rosenberg and J. Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Confer-*

- ence on *EMNLP-CoNLL*, pages 410–420, Prague, Czech Republic, June 2007. Association for Computational Linguistics.
- [64] M. T. Schultheis and A. A. Rizzo. The application of virtual reality technology in rehabilitation. *Rehabilitation psychology*, 46(3):296, 2001.
 - [65] N. E. Seymour, A. G. Gallagher, S. A. Roman, M. K. OBrien, V. K. Bansal, D. K. Andersen, and R. M. Satava. Virtual reality training improves operating room performance: results of a randomized, double-blinded study. *Annals of surgery*, 236(4):458–464, 2002.
 - [66] X. Shen, B. Tan, and C. Zhai. Implicit user modeling for personalized search. In *CIKM '05*, pages 824–831, 2005.
 - [67] W. R. Sherman and A. B. Craig. *Understanding virtual reality: Interface, application, and design*. Elsevier, 2002.
 - [68] P. K. Shivaswamy and T. Joachims. Multi-armed bandit problems with history. In *AISTATS '12*, 2012.
 - [69] M. D. Smucker and C. L. Clarke. Time-based calibration of effectiveness measures. In *SIGIR'2012*, pages 95–104. ACM, 2012.
 - [70] E. J. Sondik. The optimal control of partially observable markov processes over the infinite horizon: Discounted costs. *Operations Research*, 26(2):282–304, 1978.
 - [71] K. Song and W. Lee. A virtual reality application for geometry classes. *Journal of Computer Assisted Learning*, 18(2):149–156, 2002.
 - [72] S. Spielberg, G. R. Molen, B. Curtis, W. F. Parkes, J. . Bont, S. Frank, J. Cohen, T. Cruise, C. Farrell, S. Morton, M. . Sydow, L. Smith, P. Stormare, T. B. Nelson, S. Harris, K. Morris, and P. K. Dick. Minority report, 2003.
 - [73] E. Stamatatos, M. Tschuggnall, B. Verhoeven, W. Daelemans, G. Specht, B. Stein, and M. Potthast. Clustering by Authorship Within and Across Documents. In *Working Notes Papers of the CLEF 2016 Evaluation Labs*, volume 1609 of *CEUR Workshop Proceedings*, 2016.
 - [74] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.
 - [75] A. R. Taylor, C. Cool, N. J. Belkin, and W. J. Amadio. Relationships between categories of relevance criteria and stage in task completion. *Information Processing & Management*, 43(4), 2007.
 - [76] J. Teevan, S. T. Dumais, and D. J. Liebling. To personalize or not to personalize: Modeling queries with variation in user intent. In *Proceedings of the*

- 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 163–170, New York, NY, USA, 2008. ACM.
- [77] G. H. Yang, M. Sloan, and J. Wang. *Dynamic Information Retrieval Modeling*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers, 2016.
 - [78] G. H. Yang and I. Soboroff. Trec 2016 dynamic domain track overview. 2016.
 - [79] G. H. Yang, I. Soboroff, and J. Frank. Trec 2015 dynamic domain track overview. 2015.
 - [80] G. H. Yang, Z. Tang, and I. Soboroff. Trec 2017 dynamic domain track overview. 2017.
 - [81] H. Yang. Browsing hierarchy construction by minimum evolution. *ACM Transactions on Information Systems*, 33(3), 2015.
 - [82] Y. Yang and A. Lad. Modeling expected utility of multi-session information distillation. In *ICTIR'2009*, pages 164–175. Springer, 2009.
 - [83] M. Yannakakis and F. Gavril. Edge dominating sets in graphs. *SIAM Journal on Applied Mathematics*, 38(3):364–372, 1980.
 - [84] S. Yuan and J. Wang. Sequential selection of correlated ads by pomdps. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 515–524, New York, NY, USA, 2012. ACM.
 - [85] T. Zarraonandia, P. Díaz, I. Aedo, and A. Montero. Immersive end user development for virtual reality. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, pages 346–347. ACM, 2016.
 - [86] S. Zhang, D. Guan, and H. Yang. Query change as relevance feedback in session search. In *SIGIR '13*, pages 821–824, New York, NY, USA, 2013. ACM.
 - [87] S. Zhang, J. Luo, and H. Yang. A pomdp model for content-free document re-ranking. In *SIGIR '14*.
 - [88] L. Zhou. A survey on contextual multi-armed bandits. *CoRR*, 2015.
 - [89] M. Zyda. From visual simulation to virtual reality to games. *Computer*, 38(9):25–32, 2005.