MODELS AND EVALUATION OF USER SIMULATION IN INFORMATION RETRIEVAL

BY

SAHITI LABHISHETTY

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois Urbana-Champaign, 2023

Urbana, Illinois

ABSTRACT

Search and recommendation are crucial parts of many applications. Additionally, assistive AI systems have become very popular with successful intelligent agent systems. Although the IIR(interaction information retrieval) systems, including conversational systems, already improve the user experience for search, recommendation and question answering, the evaluation of such systems still has many challenges. For example, how to compute the overall utility of a system in helping a user in achieving their goal? How to compare different IIR systems? How to perform A/B testing that is reproducible, robust and less risky to online user experience? User simulation enables controlled and reproducible experiments and at the same time can simulate user interaction with an IIR system and can evaluate the overall effectiveness of an IIR system. User simulation in information retrieval (IR) aims to develop user models to simulate how a user interacts with an IR system. It involves simulation of user actions, behavior or decisions in a search process, like query formulation, click simulation, and so on. User simulation in IR has many applications like offline (without real users) evaluation of interactive IR systems, generating synthetic data to train IR models, especially for training reinforcement learning models, and modeling user behavior for analysing search behavior.

However, search user simulation is quite challenging. The existing models for simulating users lack interpretability; nor can they model a user's cognitive state. Most user simulation models do not leverage user search log data to learn better models from real user search sessions. Interpretability is needed to model meaningful variations in user behavior and thus simulate user actions corresponding to different types of user behavior. It is important to model a user's cognitive state to build a more generalized formal model because user actions are guided by the latent user cognitive state which constitutes user's knowledge, information need and search behavior characteristics. It is also challenging to utilize the search log data of users to build advanced simulation models. Another difficult challenge that has not yet been addressed in the previous work is how to assess the reliability of a user simulator to evaluate IIR systems.

My research aims to address the challenges in both modeling and evaluation of user simulation. To address the challenges in modeling users, I have studied how to develop new user models that are both interpretable and can model a changing user's cognitive state for user simulation in both Web search and E-commerce search scenarios. User search logs have rich information about users which can be used for building better user models. Therefore, we also propose a supervised user model based on imitation learning which can learn from large-scale search logs to simulate different user actions. To address the limitations in the evaluation of user simulation, we propose a novel evaluation framework that evaluates the reliability of a user simulator where the framework does not necessarily require real user data. Specifically, the following contributions are made towards the thesis:

1. We propose a new user model called **CSUM (Cognitive State User Model) for E-commerce search that models a changing user's cognitive state** and is parameterized meaningfully such that the parameters correlate with different user behavior.

2. Query simulation is a critical component of the user simulation. We propose a novel unified **Precision-Recall-Effort (PRE) optimization framework for simulating query formulation and reformulation** which is applicable for modeling both Web search and E-commerce search users.

3. Search logs contain rich information about user search actions and behavior, and also enable modeling users with a wide range of information needs. We propose a novel **Imitation Learning based User Model (ILUM) which is a supervised user simulation model based on imitation learning** to learn from the search logs. The ILUM model learns to simulate different user actions along with deciding what action to take next. It simulates user actions based on all its previous actions and the given user task/information need.

4. We address some of the challenges in the evaluation of user simulation by proposing a novel **Tester-based evaluation (TBE) framework to evaluate the reliability of a user simulation for comparing IIR systems**. The advantage is that the framework does not necessarily require real user data to evaluate the simulator and it aims to evaluate the predictive validity of a simulator. We further **extend the TBE framework by proposing Reliability Aware Tester-based evaluation (RATE) framework** to address the drawbacks of the TBE framework.

We propose an optimization framework for the query simulation and cognitive state models of the user during the search process through PRE and CSUM models, respectively. Both models are interpretable and can be varied in order to simulate different user behaviors. PRE simulates both initial query formulation and subsequent reformulation in a uniform manner and serves as a roadmap for the systematic exploration of many new specific query simulation models and algorithms.

However, PRE and CSUM models cannot learn search patterns from user search sessions. Thus, we also propose a data-driven approach using search logs to build a user simulation model based on imitation learning which we refer to as ILUM((Imitation Learning based User Model) model. The ILUM model can be trained to simulate a complete user model, including different user actions and decisions during the search. The ILUM model can learn complex search patterns, but unlike PRE and CSUM models it lacks interpretability, in that the model cannot be varied meaningfully to simulate different types of users and information needs for generating search sessions. As the ILUM model learns from all sessions together, it learns to simulate an average user in search, whereas PRE or CSUM models can simulate a specific type of user. An interesting research focus could be to build a data-driven user simulation model that can meaningfully simulate variation in search behavior for different user types and information needs.

Finally, we propose a novel evaluation framework, RATE, for evaluating user simulation models in terms of reliability for comparing IR systems. The advantage of RATE is that it does not necessarily need real user search data to evaluate a simulator and can complement other evaluation metrics. One of the pivotal future works for user simulation in IR would be to develop an evaluation platform with many user simulators that is available for the research community to utilize in the evaluation of IR systems or other applications.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor and chair of my committee Prof. ChengXiang Zhai for his invaluable advice, contribution, patience and unparalleled support during my PhD study. This endeavour would not have been possible without the help of my defense committee, Prof. Hari Sundaram, Prof. Kevin C.C. Chang, and Dr. Vanessa Murdock, who generously provided their knowledge and expertise. I am also extremely grateful to my internship mentors and supervisors Dr. Vanessa Murdock, Dr. Min Xie and Dr. Suhas Ranganath for their helpful advice and profound belief in my work. I would like to extend my sincere thanks to all my team members during my internships for their writing help and contribution to the research projects. I am also grateful to the Carl R. Woese Institute for Genomic Biology (IGB) for funding my research assistantship. Many thanks to the TIMAN labmates for their feedback on my research and their contribution. Finally, I would be remiss in not mentioning my parents, my spouse, other family members and friends. Their belief in me has kept my spirits high during this journey, and without their understanding and encouragement, it would not have been possible to complete my study.

TABLE OF CONTENTS

Chapter 1: INTRODUCTION

Information access systems are being advanced greatly with recent developments in the fields of artificial intelligence, deep learning, machine learning and natural language processing [1, 2]. Search and recommendation are a crucial part of many applications and products utilized by billions of users every day. Additionally, assistive AI systems have become very popular with successful intelligent agent systems like Alexa, Google Home and Siri accompanied by increasing research advances in conversational search and recommendation systems [3, 4]. Many other interactive information retrieval (IIR) systems are also being used to satisfy the long-term search needs of the users.

Although the IIR systems, including conversational systems, are already very useful in assisting users and improving user experience, evaluation of such systems still has many challenges. For example, how to compute the overall utility of a system in helping a user in achieving their goal? How to compare different IIR systems? As users have significant variance in their behavior, it is also important to know how to evaluate an IIR system in different aspects of its utility to different groups of users. Some of the popular evaluation techniques are using standardized test collections [5], user studies [6] and online evaluation [7] like A/B tests. The standardized test collections are good for reproducible experiments but they are static and thus do not evaluate the interaction with the user correctly. Utilizing real user studies or online evaluation will reflect the utility of an IIR system, but it is expensive and does not allow for reproducible and controlled experiments. Another challenge of online A/B testing is that there is a risk of exposing users to low-performing algorithms during the online evaluation. Thus simulating real users offers a good solution for the evaluation problem. User simulation enables controlled and reproducible experiments and at the same time can simulate user interaction with an IIR system and can evaluate the overall effectiveness of an IIR system.

Simulation is not new to the IR evaluation method [8, 9]. Cranfield paradigm [10] is one of the most popular evaluation techniques in IR and it is the basis for many standardized test collections including TREC tracks (Text REtrieval Conference) [11]. The Cranfield paradigm is also a form of user simulation with an overly simplified notion of the user. However, User interaction is completely abstracted in this paradigm and it makes unrealistic assumptions of the user which is not suitable to reflect the utility of an IR system in practice [12]. Thus simulation of real users is required to be able to evaluate the effectiveness of the IIR systems with reproducible experiments [13]

Simulation generally implies imitating actions of a real-world phenomenon as described in [12]. What does it mean by user simulation in information retrieval? User simulation in information retrieval (IR) aims to develop user models to simulate how a user interacts with an IR system. It involves simulation of user actions, behavior or decisions in a search process. For example, with a traditional search engine interface, a search process starts with a user making a query or request and the underlying IR system retrieves a set of results for that query. The user may skip or click on a result, and after examining the clicked results the user may choose to come back to the result page, click on another result or reformulate a query or end the session. If the user does not choose to end the session it continues with the same actions as before. This search process/session involves many user actions like initial query formulation, query reformulation, clicks, the decision to end the session, and the decision to click more results or reformulate a query. In the case of the conversational search process, the user actions may involve question formulation to the system and answering questions of the system to complete a task. Thus simulating users constitutes simulating all such user actions. Overall, the simulation model should simulate how a user interacts with a given IR system.

The underlying model of a user simulator can have interpretable parameters that may meaningfully correlate with different user behavior characteristics. The model can be used to simulate variations in users or search constraints, like simulating expert vs. novice users, exploratory search vs. specific search, more vs. less patience/time, and so on, by varying the parameters. It can be used to evaluate IIR system performance with different search constraints and different types of users. Thus in addition to enabling reproducible experiments, one of the major advantages of simulation is to be able to perform controlled experiments and even evaluate extreme scenarios like long-tail queries and infrequent information needs.

User simulation in IR has many applications in addition to offline (without real users) evaluation of interactive IR systems [14, 15] like generating synthetic data to train models, especially for training reinforcement learning models [16, 17, 18], modeling user behavior for analysing search behavior [19, 20, 21]. User simulation models for simulating queries have been used for evaluating relevance feedback [22], query suggestions [23], and click models have also been utilized for implicit and explicit relevance feedback [24, 25, 26]. Click simulation models are used in training online learning to rank models [27, 28]. In addition to search, user simulation is also studied in the domain recommendation systems [29]. Some of the most recent works in user simulation primarily involve evaluating conversational search and recommendation systems [15, 30, 31, 32, 33]. An interpretable parameterized user simulation model where the parameters meaningfully correlate with user behavior can be used to analyse real user search log sessions, and can provide insight into user search patterns or interaction patterns which can be used to improve an IR system [34]. A formal model of a user is also essential to optimizing any interactive retrieval algorithms since the objective function to be optimized by a search engine in interactive retrieval must include a mathematical description of the user that it attempts to interact with.

However, search user simulation is quite challenging because the internal decision process of a user when interacting with a search engine cannot be observed and the user behavior varies significantly across users as well as in different search contexts. Specifically, the existing models for simulating users lack interpretability; nor can they model a user's cognitive state. A cognitive state can be defined as including knowledge and information need of the user relevant for the search. The user actions are guided by the user's knowledge, information need and search behavior characteristics, thus it is important to model a user's cognitive state to build a more generalized and effective user model to simulate user actions. This thesis aims to address the challenges in both modeling and evaluation of user simulation. To address the challenge of modeling users, I have studied how to develop new user models that are both interpretable and can model a changing user's cognitive state for user simulation in both Web search and E-commerce search scenarios. User search logs have rich information about user search actions and behavior which can be used for building better user models. Thus, we propose a supervised user simulation model based on imitation learning to leverage large-scale search log data and learn from a wide range of information needs. Finally, another difficult challenge that has not yet been addressed in the previous works is how to assess the reliability of a user simulator to evaluate IIR systems. As a result, we do not yet have an established benchmark of user simulators that can be used by researchers for IIR evaluation. To address the limitations in the evaluation of user simulation, we propose a novel evaluation framework that evaluates the reliability of the user simulator where the framework does not necessarily require real user data.

Specifically, the following contributions are made towards the thesis:

1. We propose a new user model called **CSUM (Cognitive State User Model) for E-commerce search scenario that models a changing user's cognitive state** and is parameterized meaningfully such that the parameters correlate with different user behavior [34].

2. Query simulation is a critical component of the user simulation. We propose a novel unified **Precision-Recall-Effort (PRE) optimization framework for simulating query formulation and reformulation** which is applicable for modeling both Web search and E-commerce search users [35].

3. Search logs contain rich information about user search actions and behavior and also enable modeling users with a wide range of information needs. We propose a novel

Imitation Learning based User Model (ILUM) which is a supervised user simulation model based on imitation learning to learn from the search logs. The ILUM model learns to simulate different user actions along with deciding what action to take next. It simulates user actions based on all its previous actions and the given user task/information need.

4. We address some of the challenges in the evaluation of user simulation by proposing a novel **Tester-based evaluation (TBE) framework to evaluate the reliability of a user simulation for comparing IIR systems** [36]. The advantage is that the framework does not necessarily require real user data to evaluate the simulator and it aims to evaluate the predictive validity of a simulator. We further **extend the TBE framework by proposing Reliability Aware Tester-based evaluation (RATE) framework** to address the drawbacks of the TBE framework.

In this thesis, we propose user simulation models for web search and e-commerce search. The proposed frameworks can be extended to apply to other domains too like literature search and more. E-commerce search is an important application of IR and E-com search has attracted much attention recently [37, 38]. In this thesis, we have leveraged the E-Com search log available to us to study user simulation. However, as our goal is to develop general user simulation models, we did not attempt to optimize user simulation specifically for E-com search but instead used the domain to evaluate the general techniques for user simulation. Specifically, we have proposed a new user model called **CSUM (Cognitive State User Model)** which is an interpretable generative model that simulates user actions in a search session when searching for a particular product. In this work, we are addressing the question, **how to model a user cognitive state in a user model simulating user interactions in E-commerce search**. Given a target product/item as input, CSUM models how the user would formulate, reformulate query as needed, and click on results through a query reformulation model and a click simulation model respectively. In the CSUM model, we propose a background state called the user cognitive state which models the information need and knowledge state of the simulator at any given time and this cognitive state is a parameter to the query reformulation and click models to simulate the respective user actions at that given time. CSUM also includes an update model so that the cognitive state can be updated over time. The parameters in the model are meaningful and interpretable that is they correspond to a certain type of user behavior.

CSUM has multiple applications. **We used the CSUM model as a tool to mine E-commerce search log data to analyse the search behavior of E-commerce users buried in the search logs**. Through our experiments, we study **whether specific**

parameters of the CSUM model indicate different user behaviors, **How does the CSUM model perform for simulating the user's next actions given the target item and query** and how to use the CSUM model to mine search patterns from E-com search logs. To understand user behavior, we fit the model to the search log sessions by estimating the model parameters for a given user and given target product. Because of its explicit modeling of cognitive state and interpretable parameters, CSUM enables us to reveal interesting user behavior patterns that cannot be extracted using the existing methods. For example, we identify that the major variations in user behavior occur in the aspect of to what extent the user wants to explore the product space. Further, a preliminary evaluation shows that the model performs well in predicting user actions (user's next query and clicks) in a search session given the target product. Therefore, CSUM can be used to simulate user actions.

Next, we extend the CSUM query formulation/query simulation model to propose a generalized framework for query formulation which is applicable for modeling users in both Web search and E-commerce search users. Query formulation is a crucial component of user simulation. While there has been much work on modeling other user actions such as clickthroughs, there are only a few works that propose methods for modeling and simulating query formulation [14, 39, 40, 41, 42, 43]. More importantly, there is not yet a general formal interpretable model for query simulation, a gap which we attempt to fill in.

Thus we study **how to model a user query formulation process**. Specifically, we propose a novel unified **Precision-Recall-Effort (PRE) optimization framework** for simulating query formulation and reformulation in which the user would generate a query to maximize the recall and precision of the anticipated retrieval results while minimizing the effort of making the query. The query formulation in the PRE framework is dependent on the (current) knowledge state of the user, which can be updated throughout the session to capture the cognitive process of learning of the user during a search session. The explicit modeling of a user's knowledge state enables the framework to naturally model the initial query formulation and subsequent reformulations uniformly as the same generation mechanism but based on different knowledge states.

The PRE framework conceptually covers the existing query simulation models as special cases. As different instantiations of the framework would lead to different query simulation models, PRE also serves as a principled roadmap for systematically exploring different models for simulating queries. We propose and study some basic strategies to instantiate the individual components of PRE and study the soundness and potential benefit of PRE. In our experiments **we have studied the soundness of our model by verifying whether the PRE framework captures necessary components of the query formulation**

process. Experiment results show that the need for optimizing both precision and recall when generating a query is indeed supported empirically. **We also evaluated the effectiveness of the PRE framework by comparing queries simulated by the instantiations of the PRE framework to the respective real user queries**.

In general, developing a formal interpretable framework for simulating user actions including the user's knowledge state has many advantages.
1) Such a formal framework can be evaluated analytically to assess its soundness since all the assumptions made about a user would be not only explicit but also formally described.
2) The modeling and updating of cognitive state (knowledge state) in such a framework enables modeling variable user behaviors with variable knowledge states (from poor to expert knowledge state) using the same interpretable mechanism, significantly reducing the complexity of the model as well as increasing the interpretability of the model.
3) The interpretability of the framework means that the parameters introduced in such a formal framework may be interpreted as representing meaningful user variations or search constraints, thus naturally tackling the challenge of modeling variable behaviors of users without requiring training data from all the users.
4) Such a framework can serve as a roadmap to provide guidelines for improving user simulation via the improvement of each component in the framework similar to PRE.

The PRE and CSUM models are interpretable and model cognitive states where the parameters can be varied to potentially simulate different types of users. While the PRE and CSUM models are interpretable, they do not leverage search log data to learn the search patterns of users in order to simulate them. The CSUM model has parameters that can be tuned using search sessions, but it cannot learn more complex search patterns to simulate a complete user model. Additionally, PRE and most of the existing user simulation models are built based on publicly available search session data, which have very limited, only hundreds, variations of user information needs. Learning from search logs has the advantage of modeling a wide range of information needs as it contains many different sessions from different users. Thus, we propose to leverage large-scale user search logs to build user simulation models. We propose a novel **Imitation Learning based User Model (ILUM) which is a supervised user simulation model based on imitation learning** that attempts to learn from the user data in the search logs. **We focus on studying a particular imitation learning (IL) method called Behavior cloning and propose a deep-learning framework for implementing it.** Additionally, we utilize an unsupervised probabilistic method adapted from one of the previous works as a reference model. We evaluate the proposed user simulation model by using it to simulate three standard search actions for E-commerce which are querying, clicking and purchasing. Most supervised user simulation models have studied only one of the user actions in the search process, but in this thesis, we attempt to learn a unified supervised model for all user actions.

Experiments show that the ILUM model performs slightly better than the unsupervised probabilistic method overall, with a statistically significant difference. We further analyze both models for predicting different actions. The imitation learning model performs better on purchase actions but is worse than the probabilistic model on querying and click actions. Further, the ILUM model, although has a lower performance to simulate the first actions (at initial timestamps) in a session it performs much better for generating consequent actions (at later timestamps) in that session based on the context of previous actions and search engine results. This shows that **the ILUM model learns to predict actions that are coherent with the previous actions and are based on the given task.** The probabilistic model has an advantage at the beginning of the session, especially in predicting query actions. The complementary benefits of the unsupervised model and the ILUM model indicate that both of them can be potentially combined for a hybrid model with even better performance. Finally, we make the first attempt to explore an imitation learning framework for modeling user search simulation leveraging search logs. The supervised ILUM model is theoretically interesting for building a user simulation model and empirically shows good performance with great potential for further improvement, including applying advanced IL algorithms like inverse reinforcement learning.

As argued earlier, one of the major applications for User simulation is the evaluation of IIR systems because user simulation can be controlled and varied systematically, and can be used to run reproducible and repeatable IIR experiments. However, for any simulator to be really useful, we must be able to evaluate its reliability since if the simulators are not reliable, the conclusions we draw from using them to evaluate IIR systems would be questionable. Thus, we address the challenges in evaluating a user simulator by studying **how to assess the reliability of a user simulator for providing an evaluation of IIR systems?**. Such an evaluation of user simulation will advance the progress in user simulation and also helps researchers to identify and use reliable simulators as a step towards utilizing user simulation for evaluation in practice.

Evaluation of user simulation itself has multiple challenges such as variance in user behavior, unavailability of ground truth, and dependency on the intended use of a user simulator. So far user simulators have been mostly evaluated by assessing how similar a simulator is to a real user using multiple strategies, including comparing simulated and real user session data [14, 39, 44, 45], comparing retrieval performance of IR systems with the real user queries and simulated queries [40, 42, 43, 46], log-likelihood and perplexity [45]. However, almost all of these evaluation methods require real user search interaction data which is expensive to acquire. Further, those evaluation methods do not directly assess whether the simulator is reliable for comparing IIR systems.

To address the limitations in the evaluation of user simulation, we propose an evaluation framework **to evaluate the reliability of a user simulator without necessarily requiring real user data**. We propose a novel **Tester-based evaluation (TBE) approach** for evaluating the reliability of user simulators. In the TBE approach, we would construct a Tester based on a set of IR systems with an expected performance pattern, and apply such a Tester to a user simulator to see if the user simulator would generate the expected performance pattern. Testers themselves can be unreliable. Therefore, we extend the Tester-based evaluation to study how to quantify the reliability of Testers by proposing a new **Reliability-Aware Tester based Evaluation (RATE) framework**. In RATE, we define the reliability of a Tester as the probability that its expected performance pattern would be satisfied when tested with a population of users. We define and formulate both the reliability of the Tester and the reliability of the Simulator such that they can be jointly learned in an unsupervised manner. We propose two formulations namely *Symmetric formulation* and *Difference formulation* for learning both the reliability of Testers and Simulators as a fixed point solution. These algorithms enable the RATE framework to not only assess the reliabilities of Testers but also directly generate reliability scores for user simulators, facilitating the use of user simulation for evaluating IIR systems.

We construct multiple Testers and apply them to a set of representative user simulators to empirically study the effectiveness of the proposed RATE framework in identifying the reliability of simulators and also Testers. Evaluation of RATE shows that the framework can meaningfully differentiate User simulators and quantify their reliabilities. RATE can be used to evaluate any simulator.

In this thesis, we propose user simulation models for web search and e-commerce search. The models are general frameworks that can be used to develop simple to complex models. All the proposed models can be considered as fundamental frameworks that can be extended and improved to be applied to search simulation in different domains like web search, e-commerce search and more. The models have different advantages and disadvantages such that they can be integrated into building an advanced simulation model and we discuss this further in the conclusion of this thesis. However, in order to derive a general framework and a computational user model that can be applied to both domains, we assume a simplified user search process for e-commerce search where the task (purchase a product) is similar to a web search (find relevant document) and the user actions are similar (query, clicks and purchase). However, some of the simplifications should be handled to specifically model e-commerce search.

In the following chapters, we described each of the works in detail. In Chapter 2, we describe a literature review of all the relevant work in user simulation, user modeling, and evaluation of user simulation. In Chapter 3, the general query formulation framework called Precision-Recall-Effort (PRE) optimization framework and in Chapter 4 the Cognitive User State Model(CSUM) for E-commerce search is described. In Chapter 5, we describe the imitation learning model for user simulation called Imitation Learning based User Model (ILUM) model to learn from user search logs to simulate a complete user model. Finally, we propose a novel evaluation framework called RATE for evaluating the reliability of user simulators.

Chapter 2: RELATED WORK

General conceptual frameworks for user modeling have been studied for a long time (e.g., [47, 48]), but they have not yet had much direct impact on the development of mathematical models, especially for user simulation. Some of the early studies about user simulation are in the context of IR evaluation [8, 9]. The SIGIR 2010 workshop on Simulation of interaction(SimInt) [12] argues that simulation has great potential in the evaluation of IIR systems, following that the recent workshop on Simulation for information retrieval (Sim4IR) [49] has summarized some of the recent progress on modeling and applications of user simulation models.

Simulation is studied for multiple real-world phenomenon like developing self-driving vehicles [50], humanoid robots [51, 52], computer games [53], human-computer interaction [54, 55] and so on. In multiple applications, simulation models are mostly trained to generate a human response given a situation or state of the environment. Simulation has also been studied for dialog systems [56, 57, 58] in order to generate conversation or a dialogue with a system mimicking human-computer interaction. One of the differences in simulating user actions in IR is that the user actions are based on different tasks or preferences, which may not be well-defined. On the other hand, user simulation in most phenomena, for example, computer games or self-driving vehicles or robots, have a well-defined task associated with them. Therefore, the ground truth for such tasks can be easily obtained. In the search process, the user actions are guided by their preferences or information, which may change over time, and the task itself may be evolved over time. Thus, it is challenging to define a user task during a session. Second, it is also challenging to define the success or completion of a task in the search session; thus, evaluating the simulator is not well-defined. Further, there is also high variation in user behavior during the search process as different users may use different strategies while searching; thus, for a given situation there may be multiple valid responses that a simulator can generate. Therefore, the ground truth for user simulation is not well-defined.

In the following, we summarize the progress and development in user simulation in IR in three aspects, 1) User simulation models and User behavior models for simulating various user actions/interaction and or analysing user behavior, 2)Different applications in which user simulation is applied, 3) Evaluation techniques to evaluate a user simulation model.

2.1 USER SIMULATION STRATEGIES

Most of the previous works focussed on simulating one of the user actions, while few of the works proposed methods to simulate all the user actions required for interacting with the given IR system. Thus, we first describe related work on the simulation of different user actions.

The click simulation model has been widely studied for various applications like evaluation of relevance feedback [22], training online learning to rank algorithms [27, 28]. Many formal models have been proposed for modeling clickthroughs(see e.g., [45, 59]). Click simulators are also learnt by training on user click logs [14, 60]. Different user behavior models were studied for click simulation [24, 26, 40, 61, 62]. In contrast, to click simulation, other user actions like query formulation and reformulation, and session-stopping behavior are relatively less studied. My research makes contributions in this direction as we have specifically proposed novel query formulation methods in our proposed user models.

Modeling session stopping behavior of a user is one of the challenging problems in user simulation and is relatively less studied. Maxwell et al. [39, 44, 63, 64, 65] has proposed various user models for stopping behavior based on heuristics and user studies.

Query simulation has been studied using different strategies [14, 39, 40, 41, 42, 43]. For example, Keskustalo et al. [42] studied five general strategies users may use to modify a previous query in order to obtain a new query. The methods proposed in Bhaskaya et al. [40, 41] and [39] are limited to using a fixed set of terms, or fixed length query, and they cannot model reformulation. In [14], the query formulation depends on the availability of user queries and user sessions for a particular topic/information need and thus is not generalizable for a new IN if the user sessions are not available. In [43], multiple query generation methods have been proposed for known-item topics, but query reformulation has not been studied.

There are two major limitations of the previous works on query formulation modeling. **First, previous works have almost all focused on specific methods for generating a query without a general framework or any discussion of the optimality of the proposed methods, making those proposed methods somewhat arbitrary. Second, none of the previous works has modeled query formulation with consideration of a user's cognitive state, an important goal that our work attempts to achieve.** Without a model of the cognitive state, previous works cannot model the initial query formulation and the subsequent reformulation in a uniform way. One might argue that having separate models for the initial formulation and reformulations later may have little practical consequence from the perspective of generating user simulation data, but we believe that this would make the formulation and reformulation models not only less interpretable but also less generalizable. As a result, such a model can only be used for modeling the users from whom we have collected data, but cannot be used to simulate unobserved users. **These are the limitations we address in our proposed query formulation models.**

Some of the previous works have studied complete user models like developing search agents to create realistic user simulations [39] and generating entire user sessions [14, 19] to create data collections.

User simulation has been studied for different problems as well for example, interacting with conversational agents by simulating questions and responses [15, 32, 33, 60, 66], interacting with dialogue systems [16, 31, 67]. User simulation is utilized for evaluating and training conversation recommendation systems. Most of the techniques involve an NLU (natural language understanding) component, response generation component and NLG (natural language generation) components. Response generation can be done either using model-based [68, 69] or an agenda-based methods [33, 70, 71]. Conversational systems are different from web and E-commerce search scenarios in that the NLU and NLG are specific to the conversational systems.

2.1.1 E-commerce search simulation

User simulation modeling is not explored in the area of E-commerce search. In E-commerce (E-comm), there are few works [72, 73] analysing queries and user behavior patterns through search logs but they haven't explored query simulation methods. Query suggestion, personalized recommendation and user behavior analysis are studied in the E-commerce domain [72] which are some of the works that are close to the user simulation models. E-commerce search simulation has advantages in that the information need is more structured however it also has challenges like search ranking can be affected by multiple factors [74, 75] and the user relevance model and actions are affected by different factors than the relevance to the product preferences like price, reviews, quantity, availability and more [38].

There are works analysing or predicting user purchase or browsing behavior [76, 77, 78, 79] in web search but these works do not generalize to the E-com search domain. The user simulation models in Web search [14, 39, 41] cannot be easily generalized to generate user interaction in E-comm. Our research contributes to this line of work in that we propose a novel cognitive user model for simulating query formulation and clicks in E-commerce search scenarios.

User behavior models have been studied to analyse search behavior. Such models inform the design of both search systems and user simulation models. Economic models of user decisions and effective search behavior models [80, 81] have been proposed by Azzopardi et al. [20, 82, 83]. Our proposed user model PRE adds to this line of work where we propose an optimization framework for the query formulation process, further, our proposed user models enable simulating variations in user behavior by varying the corresponding parameters in the model. We propose a cognitive user state model (CSUM) that can be used to mine E-commerce search logs to identify interesting user behavior patterns.

2.1.2 Supervised learning for training user simulators

Supervised learning approaches can be used to train user simulator models using the real user search logs as training data although there might be a few challenges. Training to simulate each and every user action requires a large amount of training data, at the same time overfitting to the training data may restrict the user simulator to only simulate the observed sequences of user actions.

Carterette et al. [14] has proposed a learning to rank framework to predict clicks and perform click simulation on a result ranked list. Zhang et al. [60] has proposed a bidirectional GRU network to train a click simulator. In both the models, the clicks are predicted for a given query and they do not consider the context of a session of the previous queries and clicks. Many click models have been designed to use supervised learning [45]. The above click models are applied for evaluation of relevance feedback, and training online learning to rank [28]. Large scale user search logs are also used for learning query simulation methods to rewrite [84] or reformulate queries [85, 86], but these methods rely on the availability of search logs.

However, a supervised user simulation model simulating different user actions is not studied in the previous works. Moreover most of the previous works do not model or evaluate using large number of information needs which limits the generalizability of the methods. We propose a supervised user model based on imitation learning referred as ILUM to learn from a large number of real user search logs, thus addressing some of the above limitations of previous works.

2.2 APPLICATIONS OF SIMULATION

Simulation has been used in various applications in evaluation of search and recommendation, it is used to evaluate relevance feedback based models [22], query suggestions [23], query expansion [87], known-item retrieval task [43, 88]. Simulation of user interaction has also been used to train reinforcement learning algorithms [18, 60]. Many recent works have applied user simulation to train and evaluate conversation search systems and conversational recommendation systems [15, 32, 33, 60, 66]. Evaluating conversational systems requires evaluation of the dialogue/conversation between the user and system, it is expensive to perform this evaluation with real users. Thus user simulation has been widely used for training and evaluating conversational search and recommendation systems.

2.3 EVALUATION OF USER SIMULATION

In addition to modeling user simulation, evaluation of user simulation is crucial to be able to practically utilize the user simulation. As described in Chapter 1, the evaluation of user simulation itself has many challenges. The evaluation of simulators has almost all been based on comparing the behavior of a simulator with that of real users using, e.g., retrieval performance [40, 42, 43, 46] or statistics of actions such as the number of queries and clicks [39, 44]. Similarly, search log data is used to compare simulated data to real user data to compute the accuracy of simulation [14, 45]. In [45], log-likelihood and perplexity are also used to evaluate click models. In [22, 23], different user simulator methods are used to study how query suggestion and relevance feedback algorithms perform with different simulator behaviors but the simulator methods themselves are not evaluated. Although these evaluation methods are meaningful, they cannot inform us of which simulators are more reliable for comparing IIR systems. In [14], the authors compared the simulated queries with real user queries in terms of their impact on ranking IIR systems, which is a step towards evaluating user simulators from the perspective of comparing IIR systems. However, this work has three limitations: 1) It requires real user session data, which are expensive to acquire, limiting its application scope. 2) It can only be used to compare user simulators in one functional aspect (i.e., query formulation). 3) The evaluation results are not interpretable in that they can only indicate which simulator is more reliable, but cannot explain why or in which way. We address all the three limitations of this previous work and propose a new evaluation framework called Reliability aware Tester-based evaluation approach.

Therefore, in our research, we propose query simulation methods that are both interpretable and model a latent cognitive state of the user. We directly tackle the problem of optimality in both initial query formulation and subsequent reformulation and propose a general framework for query formulation and reformulation with well-articulated optimality criteria. As we will show, the proposed framework Precision-Recall-effort (PRE) optimization can cover all the major techniques and ideas proposed in the previous works (e.g., [14, 39, 43]) as special cases, which is additionally useful to compare them systematically both analytically and experimentally. We propose a novel cognitive state user model (CSUM), a conditioned generative model that can be used to both analyse user behavior and also simulate user queries and click actions given a target product as information need. CSUM models a cognitive state of the user which generates the user actions and the model includes meaningful parameters that correlate with user behavior which is used to discover interesting search behavior patterns from E-commerce search logs. We address the limitation in learning from search logs and the modelling and evaluation using limited number of information needs by proposed a supervised user simulation model based on imitation learning ILUM which can learn from large scale search log data. We address some of the limitations in the evaluation of user simulation and propose a new evaluation framework called Reliability aware Tester-based evaluation (RATE) approach that enables systematic interpretable evaluation of the multi-aspect reliability of user simulators for comparing IIR systems, that also generalizes what is done in [14]. The approach also does not require the use of real user data, though, of course, a Tester can also be constructed based on real user data whenever they are available.

Chapter 3: PRE: A PRECISION-RECALL-EFFORT OPTIMIZATION FRAMEWORK FOR QUERY SIMULATION

In this chapter, we study how to simulate query formulation and propose a novel optimization framework for simulation query formulation and reformulation uniformly. The optimization framework is used to instantiate different query simulation models and is flexible to develop improved query simulation models. We proposed this framework for queries in a web search scenario but it can be adapted to apply to other domains as well.

3.1 INTRODUCTION

Query formulation is an important user action and is challenging to simulate because the space of possible queries a user may simulate for an information need is huge. Few works propose query simulation methods but they do not consider a user cognitive state (which includes user knowledge) as a component of the user simulation model. Further, the existing work has focused on specific methods for generating a query without a general framework or any discussion of the optimality of the proposed methods, making those proposed methods somewhat arbitrary [14, 22, 39, 43].

In this chapter, we study how to model the user's query formulation process, we propose a formal optimization framework that generates a query conditioned on the current knowledge state of the simulator and the knowledge state is updated for simulating the next query during query reformulation, therefore the query formulation and reformulation are modelled uniformly in our work. Our proposed framework can cover existing query simulation methods as special cases and can be used to make many new query simulation methods by instantiating each component of the framework in different ways. We study the effectiveness of the new instantiations and analyse the framework for its soundness and influence of different components on the performance.

A formal model of a user is essential for optimizing any IIR algorithms since the objective function to be optimized by such an algorithm must include a mathematical description of the user that it attempts to interact with. Moreover, an interpretable parameterized user simulation model where the parameters meaningfully correspond to different user behaviors can also be used as a tool to mine real user logs to identify interesting user search behavior patterns [34], in addition to simulating different user search behaviors.

However, formally modeling and simulating user interactions is very challenging for the following reasons.

First, the interaction process of a user with an IR system is an unobservable complex cognitive process where the user's knowledge and information need can also be frequently updated [47, 48]. Existing research in cognitive science and search user studies only provides a limited understanding of this process. We thus do not have a clear theoretical basis for modeling users mathematically. Second, user interaction has a lot of variance; users with the same information need may show different types of querying and clicking behaviors. Thus a simulator should also be able to vary in a meaningful way to simulate different kinds of user behaviors, such as variations in their knowledge background, patience, or tradeoff between effectiveness and effort. Further, the simulator should also adapt to different information needs. Finally, evaluation of the user simulation model also poses multiple challenges [36]. While there has been some progress in evaluating user simulators empirically (e.g., the Tester-based evaluation approach [36, 89] and the multi-dimensional evaluation framework [14, 90]), there is a lack of progress in evaluating the soundness of the model behind a user simulator, which is primarily because the existing user simulators do not clearly articulate the assumed generative process that an actual user uses in generating the observed search behavior.

Due to these challenges, progress in the research of user simulation has been slow, especially in developing formal models that can provide an interpretable explanation of how users formulate queries. While many models and methods have been proposed for modeling clickthroughs (see e.g., [45, 91, 92, 93]), only a few methods have been proposed for query simulation [14, 39, 40, 41, 42, 43, 90].

There are two common deficiencies in all the existing approaches to simulating user query formulation. First, the existing approaches do not explain how or whether the simulation algorithm is based on an assumed generative process that an actual user may follow for formulating a query, making it hard to extract any meaningful hypothesis about the whole process of the user query formulation from such algorithms. Moreover, all the existing methods with few exceptions [34, 39, 94] do not incorporate a user knowledge state as an underlying variable for the query formulation process, even though a query formulated by a real user is clearly influenced by their knowledge. Second, although most algorithms adopted an implicit optimization framework to generate a query that is "optimal" by some criteria, the objectives to be optimized are often not explicitly articulated, nor can they be easily interpreted as meaningful objectives that a user could conceivably optimize when formulating a query, again making it hard to interpret those algorithms from the perspective of simulating real users.

One may argue that as long as the synthetic queries generated by a query simulator are similar to a real user's queries or give similar performance, the interpretability and how realistic the simulator is in terms of simulating the actual query generation phenomenon are not very critical, thus developing a highly interpretable simulator may be only theoretically interesting. However, we would argue that an interpretable simulator is not only theoretically interesting but also is of practical importance. When a simulator only has a high empirical validity but is not theoretically sound, it would have limited value in applications because its generalization capacity of generating realistic queries unseen in the training data set is questionable; yet, being able to generate unseen realistic queries is precisely why we need a simulator in the first place. Further, the best way to evaluate simulated queries is already an open challenge raising the question of the reliability of pure empirical validation.

In this chapter, we address the above limitations of the previous works by proposing a novel interpretable Precision-Recall-Effort (PRE) optimization framework for simulating query formulation and reformulation. In the PRE framework, we make the following explicit assumptions (hypotheses) about how a user formulates a query: 1) A user is assumed to generate a query to maximize both the recall and precision of the anticipated retrieval results while minimizing the effort of making the query. 2) A user would attempt to estimate the recall and precision of the anticipated retrieval results based on the (current) knowledge state of the user. The knowledge of a user includes, as a minimum, knowledge about how the search engine would process a query (e.g., retrieving documents matching keywords in the query), and knowledge about the relevant and non-relevant (distracting) information (e.g., terms that may occur in relevant or non-relevant information items). 3) A user's knowledge state would be updated throughout a search session as the user learns during the search session; thus, reformulation of queries can be assumed to follow the same initial query formulation process but with an updated knowledge state.

The precision and recall are estimated using a Model of Precision (Prec) and a Model of Recall (Rec); both can be defined based on probabilistic models conditioned on the knowledge state of the user.

As an interpretable optimization framework for query formulation with explicit objectives, the proposed PRE framework has the following benefits that cannot be offered by existing approaches to query simulation:

1. Explicit assumptions and hypotheses: PRE can be evaluated analytically to assess its soundness since all the assumptions made about a user would be not only explicit but also formally described. User studies can potentially help validate or further improve those assumptions, advancing our understanding of how users formulate a query.

2. Simulation of variable behaviors of users: The interpretability of the framework means that the parameters introduced in PRE can be interpreted as representing meaningful user variations, thus naturally tackling the challenge of modeling variable behaviors of users without requiring training data from all the users. For example, the influence of the three objectives (Precision, Recall, Effort) in the optimization objective function can be controlled

flexibly by parameters; varying these parameters enables to simulate plausible different querying strategies that users may adopt without requiring additional user data for training.
3. Roadmap for systematic exploration of query formulation simulators: PRE can serve as a roadmap for systematic exploration of specific query simulation methods. Different ways to instantiate the framework would lead to many new query simulation methods. It further enables examination and comparison of the proposed query simulation methods in existing literature analytically as PRE covers existing approaches as specific instantiations.
4. Extensibility with other user modeling components: PRE can be extended by plugging in any existing models for modeling a user's knowledge state [94] into the PRE framework, incorporating other interpretable models related to a user's query formulation (e.g., economics models [95]), and combining it with other models of search user actions like click models.

As mentioned above, the PRE framework can be instantiated in many ways resulting in different query simulation methods for simulating queries. While a full exploration of this potential of PRE is out of the scope of this work, we explore and study some basic strategies to instantiate the individual components of the PRE framework to derive multiple query simulation algorithms. These simulation algorithms are used to study the soundness of the design of PRE, that is, whether the high-level objectives to optimize and the uniform modeling of both initial and subsequent query formulations are empirically supported. Our experiment results show that optimizing both precision and recall when generating a query is indeed necessary in that optimizing only one of them has consistently resulted in lower performance compared to optimizing both together. This suggests that the objective function of PRE is reasonable, and conceptually the query formulation problem can be reduced to choosing a query to maximize both precision and recall. The results also show that the performance patterns of different query formulation methods in initial query formulation are similar to their patterns observed for query reformulation, suggesting that the uniform query formulation mechanism adopted in PRE is reasonable and it is possible to improve query formulation in a general way to impact both initial and subsequent query formulations. However, we observed that improving precision and recall does not have an additive effect on overall performance improvement, suggesting that there is a potential interaction between precision and recall models that needs to be further studied.

The main contribution of this chapter is the introduction and study of a novel interpretable optimization framework PRE, which is based on hypotheses explaining how real users formulate their queries. The framework formally connects the query formulation process with the knowledge state of a user, simulates both initial query formulation and subsequent reformulation in a uniform manner, and serves as a roadmap for systematic exploration of many new specific query simulation models and algorithms. Query simulation algorithms can be used as part of a user simulation model for evaluating or optimizing IIR system, for generating synthetic data collections [14, 63, 96] and also individually for evaluating query suggestion algorithms [23].

3.2 RELATED WORK

User simulation has been studied in the past from different perspectives. For example, early studies focused on using user simulation for IR evaluation [8, 9]. It also has been used for analysing user search behavior patterns [34], training reinforcement learning algorithms [16], and recently for evaluating conversational search systems [32, 33]. While many click modeling approaches are proposed [45, 91, 92, 93], less work has been done on query modeling [14, 39, 40, 41, 42, 43, 90]. Some of the previous works attempt to simulate all user actions (e.g., [14, 34]).

Multiple query simulation methods have been proposed in the previous works, but no previous work has attempted to propose a query simulation method based on an explicit hypothesis about the process that a user may have used to formulate a query (especially the initial query), which is achieved in our proposed PRE framework. For example, query modification strategies have been studied and utilized in many methods (see, e.g., [19, 22, 23, 39, 41, 42, 90]). While those query modification strategies are realistic user strategies, the previous work has not attempted to explain why a user has chosen a particular strategy which may also be different at different times. The PRE framework offers a potential explanation of such behavior from the perspective of a user's attempt to optimize precision and/or recall while minimizing effort (editing a previous query involves less effort than generating a new query from scratch). Most existing query simulation algorithms are based on a language model of information needs (e.g., [14, 22, 39, 43, 90]). Most early simulators [43] are not parameterized and thus cannot simulate variable user behavior or preferences. A recent work [90] overcame this limitation by adapting the query change model [97, 98] for query simulation and enabling limited user variation by introducing parameters to denote user preferences like preference to retain previous query terms or preference to stick to the topic. Almost all of these methods can be covered as special instantiations of the PRE framework. Large scale user search logs are also used for learning query simulation methods to rewrite [84] or reformulate queries [85, 86], but these methods rely on the availability of such data. In the PRE framework, the model of recall and precision can either be computed using TREC collections as pursued in our experiments or be trained using user search log data if available.

Multiple studies have investigated how learning occurs during the search process [99, 100, 101] and how different user characteristics may have an impact on learning [102, 103] including knowledge of the users. However, all the query simulation methods, with only a few exceptions ([14, 39, 94]) have failed to capture the impact of a user's knowledge state on query formulations. The PRE framework established a general probabilistic model to connect query formulation with knowledge state, enabling uniform modeling of initial query formulation and all subsequent query reformulations. The specific user knowledge state and update model we have explored in this study is similar to the strategy proposed by Maxwell et al. [39].

A significant limitation of all the existing work on query simulation is that the simulation methods are not grounded on any explicit hypothesis regarding how the real users actually formulate queries. Also, although the query scoring is optimized in most of the existing approaches, it is not explicitly explained what objectives are optimized, making it extremely hard to assess whether the objectives optimized by the existing algorithms actually reflect what a real user might aim to optimize while formulating a query. There are descriptive models of search behavior and search process [47, 104, 105, 106, 107, 108] but they are not mathematical models that can be applied for simulating user behavior or generating queries. For example, the microeconomic theory has been used to study the effort and effectiveness of querying and browsing [95, 109]. Our framework is also grounded on similar principles where we optimize the effectiveness and effort of queries for query formulation, and the microeconomic models can be potentially incorporated into our framework as additional constraints for optimization.

3.3 THE PRECISION-RECALL-EFFORT QUERY SIMULATION FRAMEWORK

3.3.1 Maximization of recall and precision

To develop an interpretable framework for query simulation, we must consider how a user might formulate a query. Logically, a user would want to generate a query that can retrieve relevant documents without retrieving any non-relevant ones, i.e., optimize the quality of the anticipated retrieval results. As Recall and Precision are two basic meaningful measures of quality of retrieval results from a user's perspective, it is reasonable to assume that a user would choose a query to maximize the expected recall and precision. However, how does a user estimate the expected recall and precision of a query?

To address this question, let's consider an example of a real query "collecting old US coins" from the TREC Session Track dataset [110], where the information need (IN) is to "Obtain information on how to start collecting old US coins." It is instructive to analyze how a user might have come up with such a query.

First, it is natural for the user to think about “US coins” as it is the general topic of the IN. We note that “US coins” might be the most popular term in the relevant documents, and a user’s tendency to use such a popular term reflects the desire to match as many relevant documents as possible (i.e., maximize recall). Thus to maximize recall, a query is chosen such that it would “match” as many relevant documents as possible, where “match” indicates that the document is likely retrieved for the query by a search engine, and we will use the word “match” in this notion throughout the chapter.

Formally, let $Match \in \{0, 1\}$ be a binary variable that denotes whether there is a match ($Match = 1$) or not ($Match = 0$) between a query q and a document d in the collection $\mathcal{C}$. We use $p(Match = 1|q, d)$ to denote the probability that there is a match between a query q and document d. Let $\mathcal{R} \subset \mathcal{C}$ be the subset of relevant documents. To maximize the recall of the anticipated results, a user can be reasonably assumed to come up with a query q that would maximize $p(Match = 1|q, d)$ for **all** relevant documents, which can be formally denoted by the following Model of Recall (Rec) component in the objective function

$$\textbf{Model of Recall} : \textbf{Rec}(q, \mathcal{R}) : \prod_{d \in \mathcal{R}} p(Match = 1|q, d). \tag{3.1}$$

Note that the conjunctive expression here (instead of a disjunctive expression) encodes the objective of finding a q that can match every relevant document in $\mathcal{R}$.

However, maximizing recall is unlikely the only objective in a user’s mind since such a query also tends to match “too many” documents, including non-relevant ones. Taking the previous example of the TREC real user query discussed earlier, the shorter query “US coins” alone might match many documents about current US coins that don’t have information about old US coins nor about how to collect them. This explains why the user has further added the phrase “collecting old” which helps make the query more specific, resulting in the final query “collecting old US coins”, which can be expected to have higher precision than the previous candidate “US coins” due to its discrimination against non-relevant documents. This example shows that in addition to maximizing recall by choosing a query that can match all relevant documents, a user may also attempt to maximize the precision of the retrieval results by choosing a query that does not match any non-relevant document.

Formally, to avoid matching non-relevant documents means to minimize the probability $p(Match = 1|q, d)$ for every non-relevant document $d \in \mathcal{C} - \mathcal{R}$, which conceptually is equivalent to maximization of precision since the precision reaches a maximum when we do not retrieve any non-relevant documents. Thus, we assume that when a user composes a query, the user would also attempt to maximize the precision captured by the following Model of Precision (Prec) component in the objective function

$$\textbf{Model of Precision} : \textbf{Prec}(q, \mathcal{R}) : \prod_{d \in \mathcal{C}-\mathcal{R}} (1 - p(Match = 1|q, d)), \tag{3.2}$$

where $\mathcal{C} - \mathcal{R}$ is the set of non-relevant documents and $1 - p(Match = 1|q, d)$ is the probability that q does not match d. Once again, the conjunctive, instead of disjunctive, relation here captures the goal of not matching any of the non-relevant document.

The Rec and Prec can be combined naturally into one single objective function to capture the objective of optimizing both recall and precision. Indeed, the product $\textbf{Rec}(q, \mathcal{R})\textbf{Prec}(q, \mathcal{R})$ is precisely the probability that query q matches every relevant document but does not match any non-relevant ones. Despite the theoretical attractiveness of using this product directly as an objective function, in reality, we often need to accommodate the inevitable tradeoff between recall and precision. Indeed, maximizing precision often means sacrificing recall and vice versa. For example, consider again the information need in the previous example with an extension, "Obtain information about old US coins, how to start collecting and selling them?". While the query "US coins" might have a higher recall but lower precision, the query "collecting and selling old US coins" may be the opposite and too specific to retrieve sufficiently many relevant documents. In some cases, increasing precision may miss to retrieve relevant results or lead to zero retrieved results if the document does not match the query completely or only matches part of the query. Thus recall and precision should be balanced to obtain satisfactory search results. The optimal tradeoff between them may depend on multiple factors, including user preference or specific information needs. For example, recall might be more important in the case of finding all literature articles to write a comprehensive survey, while precision may be more important if a user would simply want to know the major events today by finding a few relevant news articles. Thus a general query formulation framework must have a precision-recall weighting parameter to enable it to be sufficiently flexible to accommodate variable tradeoffs between precision and recall. In the proposed PRE framework, we will thus choose a query to maximize the following objective of a weighted combination of Rec and Prec

$$g(q, \mathcal{R}, \alpha) = \alpha \log \textbf{Rec}(q, \mathcal{R}) + (1 - \alpha) \log \textbf{Prec}(q, \mathcal{R}), \tag{3.3}$$

where $\alpha \in (0, 1)$ is an interpretable weighting parameter to indicate the importance of recall relative to precision, which we can vary to simulate different user behaviors or user needs.

3.3.2 Modeling effort

During the search process, the user also naturally wants to minimize the effort. The effort of the user can also be understood as the time spent by the user for performing an action; however, the effort formulation need not necessarily be in terms of seconds or time units. Thus, the effort of a user during a search session is in multiple aspects, for example, 1)making a query, 2)examining the search results for the relevant results, and 3)other browsing actions. To explain the query formulation, we assume a simple interface that considers the standard actions, which are query and click actions. While formulating a query, the query formulation process directly indicates the effort spent on making the query, and it also influences the search results and, therefore, indirectly affects the effort/time spent on examining the search results for relevant results. The effort for making a query is an immediate effort, while the effort for examining the search results is the delayed effort of the user during the query formulation process.

The effort for examining the results can be reduced by an optimal query that brings relevant results at the top of the ranked result. The maximization of precision and recall of the query increases the benefit to the user to find more relevant information to satisfy their information need. At the same time, maximizing precision and recall will also optimize ranked results, which in turn also reduces the time spent or the effort of the user for examining the search results.

The second type of effort is the effort for making the query which constitutes the time and cognitive load of the user to make a query. In general, the effort required for formulating a query q, which we denote as $\mathbf{E}(q)$, can be modelled utilizing two types of work.

1. Cognitive Effort: The first is the cognitive effort required to think about the query words; a query with rare or difficult terms/words may be assumed to require more cognitive load and time from the user and thus have a higher effort.

2. Physical Effort: The second is the effort needed to physically communicate the query to the search engine, e.g., the effort required for typing in the query; shorter queries generally take less time/effort, and longer queries take more time/effort. The effort spent to type in the query can be based on the length of the query, which can be measured in terms of both the words and the letters in the query.

The effort for making a query $\mathbf{E}(q)$ can be used either as an objective to be minimized or as a constraint like a max limit on query length ($|q| < l$), which is again equivalent to an objective which is 0 if $|q| < l$ and ∞ if $|q| >= l$.

Finally, the two types of effort are captured by our optimization constraints. The maximization of precision and recall optimizes the ranked results, thereby indirectly affecting the effort on examining results, and the query effort E(q) directly determines the effort for making the query.

Our proposed multi-objective optimization framework can also be correlated with an economics model for IR studied in previous literature [109, 111]. The maximization of Precision and Recall models maximize the amount of relevant information gained by the user, thus indicating the benefit gained by the user while indirectly reducing the cost/time spent by the user on the search result page. Second, the effort for making a query indicates the cost/time spent on querying.

An effective query that maximizes precision and recall would reduce the effort while examining the search result but often requires more time/effort to formulate that query, thus higher $E(q)$. On the other hand, less effort while making the query may result in an ineffective query which implies more effort while examining the search results. We consider this tradeoff between the two types of effort by introducing a weight on the effort value $E(q)$ in the optimization function. Additionally, as described earlier in Section 3.3.1, the general query formulation framework must have a precision-recall weighting parameter to enable it to be sufficiently flexible to accommodate the variable tradeoff between precision and recall.

In the proposed PRE framework, we will thus choose a query to maximize a weighted combination of Rec and Prec, along with weight on effort for making the query. Thus, in general, the query formulation process is a multi-objective optimization process involving three (potentially conflicting) objectives: (1) Model of Recall: $\mathbf{Rec}(q, \mathcal{R})$; (2) Model of Precision: $\mathbf{Prec}(q, \mathcal{R})$; and (3) Effort for making the query: $\mathbf{E}(q)$, leading to the following general Precision-Recall-Effort (PRE) optimization framework for query formulation and reformulation.

$$\begin{aligned} q^* &= \arg\max_q g(q, \mathcal{R}, \alpha) - \lambda \mathbf{E}(q) \\ &= \arg\max_q \alpha \log \mathbf{Rec}(q, \mathcal{R}) + (1-\alpha) \log \mathbf{Prec}(q, \mathcal{R}) - \lambda \mathbf{E}(q), \end{aligned} \tag{3.4}$$

where $\lambda > 0$ is an interpretable parameter controlling the tradeoff between query quality and user effort.

The PRE framework provides a general theoretical framework for simulating how a user formulates a query based on the assumption that the user has the following knowledge: (1) Knowledge about the collection of information items $\mathcal{C}$; (2) Knowledge about relevant item set $\mathcal{R} \subset \mathcal{C}$; (3) Knowledge about how to estimate $p(Match = 1|q, d)$, i.e., how a search engine works. In reality, the users may not have accurate knowledge about any of these (if they did, they would be able to formulate a perfect query), especially in the initial stage of the search. As the user interacts with a search engine more, the user may gain more knowledge in all the three areas above. The PRE framework enables us to model the cognition process of a user during the search process by accommodating updating of any of the knowledge over time. In this way, the framework models the initial query formulation and subsequent reformulations in a uniform way with the difference only in the assumed knowledge of the user at the time of formulating a query. Next, we discuss how to model a user's knowledge state in detail.

3.3.3 Knowledge state

The user's cognitive process during the search process is complex and unknown. The user knowledge during the search process can be potentially modelled as multiple components. We describe some of the knowledge components in the following. As the search is initiated due to an information need, one of the knowledge types is the knowledge of the user regarding their information need; this knowledge is generally partial/in-complete because the knowledge gap of the user gives rise to the information need, without the knowledge gap there would be no information need as argued by ASK theory [47]. Thus, the knowledge of concepts related to or similar to the concepts involved in the information need could be represented as one of the knowledge components of the user during a search. This knowledge can be represented in different ways, such as latent concept/topic level representation [112, 113] or a conceptual graph [114]. Second, in addition to concept-level knowledge, vocabulary knowledge or word-level knowledge can also be another component. The vocabulary knowledge implies the words known to the user, which is useful, especially for the query formulation process. For example, a user may be an expert and have complete knowledge of the vocabulary regarding the IN, or they may have poor knowledge of the vocabulary. Without the knowledge of words, it is hard to formulate a query, even if the user is familiar with the corresponding concepts. The vocabulary knowledge may also include knowledge of which words correlate to which concepts and how popular or frequent the word is in the concept. Third, general knowledge or background knowledge, for example, which words are more common in other documents or information objects or stopwords and so on. Finally, the knowledge of the user regarding how the search engine works is also a knowledge component involved in the search. This knowledge is important to make all decisions in search, formulating a query such that relevant results are retrieved at the top, interacting with the search engine or browsing the search results for clicks.

The above-described knowledge components can also be varied from rich to poor knowledge in order to simulate variations in user behavior during a search, thus simulating different types of users leading to variations in the simulated sessions. The knowledge components can be varied from a low knowledge state to a high knowledge state. For example, a user may have high to low knowledge of the vocabulary related to the information need or high to low knowledge of the frequency of words in the relevant information. For example, the user may have high to low knowledge of the concepts related to the information need. Therefore, such variations can be modelled in order to simulate different types of sessions with different types of knowledge states of the user.

In the PRE framework, we model the word-level knowledge component, i.e., the knowledge of the likelihood of words in the relevant and irrelevant information. We also consider a fixed set of vocabulary for the entire session. When we apply the PRE framework to simulate a user, the whole optimization problem must be framed in the context of a user and the user's knowledge, which we denote by K. While $\mathbf{Rec}(q, \mathcal{R})$ and $\mathbf{Prec}(q, \mathcal{R})$ capture user's objective to optimize, they are computationally complex. It is unlikely that a user would be able to keep track of all the relevant and non-relevant documents cognitively and follow the exact formulas to do the computation. Since how exactly a user stores knowledge about relevance is unknown, as an initial exploration of PRE, we start with the simplest model of a user's knowledge state, where we assume that the user would accumulate and summarize the knowledge about relevant documents in $\mathcal{R}$ with an aggregated prototype relevant document $\mathcal{R}_k$ and that about non-relevant documents in $\mathcal{C} - \mathcal{R}$ with an aggregated prototype non-relevant document $\bar{\mathcal{R}}_K$; this way, the user would only need to keep track of these two prototype documents for representing relevant and non-relevant information, respectively, and the knowledge state K is mainly composed of two prototype documents $\mathcal{R}_K$ and $\bar{\mathcal{R}}_K$, i.e., $K = \{\mathcal{R}_K, \bar{\mathcal{R}}_K\}$.

Under such a prototype document assumption, we have $\mathcal{R} = \{\mathcal{R}_K\}$ and $\mathcal{C} - \mathcal{R} = \{\bar{\mathcal{R}}_K\}$, both containing just one (prototype) document, thus the product is no longer needed in the definitions of Rec and Prec, leading to the following knowledge state-dependent models of recall and precision,

$$\begin{aligned} \mathbf{Rec}(q, \mathcal{R}_K) &: p(Match = 1 | q, \mathcal{R}_K), \\ \mathbf{Prec}(q, \bar{\mathcal{R}}_K) &: 1 - p(Match = 1 | q, \bar{\mathcal{R}}_K). \end{aligned} \tag{3.5}$$

Adding the effort model $\mathbf{E}(q)$, we obtain the PRE framework,

$$q^* = \arg\max_q \alpha \log \mathbf{Rec}(q, \mathcal{R}_K) + (1 - \alpha) \log \mathbf{Prec}(q, \bar{\mathcal{R}}_K) - \lambda \mathbf{E}(q). \tag{3.6}$$

It is reasonable to assume that initially, a user's knowledge about $\mathcal{R}_K$ is mainly based on a brief description of the information need since the user has not yet seen any relevant document, and a user's knowledge about $\bar{\mathcal{R}}_K$ can be assumed to be based on a general sense about the content in a collection $\mathcal{C}$. As a user interacts more with a search engine, the user would be able to see more examples of relevant and non-relevant documents and thus update the user's knowledge by adding more information about relevant documents to $\mathcal{R}_K$ and more information about non-relevant documents to $\bar{\mathcal{R}}_K$. The exact form of updating depends on how exactly the user stores the knowledge about $\mathcal{R}_K$ and $\bar{\mathcal{R}}_K$. We will further discuss this issue in Section 3.4 under the assumption that the user would store the knowledge in the form of a unigram language model, i.e., the probability that a word w is seen in a relevant prototype document ($p(w|\mathcal{R}_K)$) or in a non-relevant prototype document ($p(w|\bar{\mathcal{R}}_K)$). The updating of a user's knowledge increases the user's capacity to potentially formulate a better query, and the PRE framework naturally captures this by simply using a more enriched knowledge state of the user for formulating a query.

The explicit connection of the query formulation (Rec and Prec) with a user's knowledge state (K) in PRE not only enables the modeling of initial query formulation and subsequent reformulations in a uniform way (as it should be), but also allows for meaningful variations of the simulated users by simulating different knowledge backgrounds of users; for example, the novice users vs. expert users can be simulated by varying how $\mathcal{R}_K$ is initialized. It also enables a natural integration of modeling query formulation with modeling the cognition of users during the search process.

So far, we have explained all the major elements in PRE, but we have not yet explained how a user might estimate $p(Match = 1|q, d)$, which is the basis for computing the models of both recall and precision in the objective function. This has to do with a user's knowledge about how a search engine works and can be potentially instantiated in many ways, which we will elaborate in the next section. The existing query formulation methods can generally be interpreted as special instantiations of the PRE framework.

3.4 INSTANTIATION OF THE FRAMEWORK

An important benefit of PRE is that by instantiating each component in PRE in different ways, we can systematically explore and study many new query simulation algorithms in the same unified framework. As an initial step, we propose some basic instantiations of PRE using statistical language models (LMs), leaving a thorough exploration as future work.

3.4.1 Conjunctive vs. Disjunctive matching

The major component that we need to instantiate is the matching likelihood $p(Match = 1|q, d)$, which models a user's assessment of whether a document d matches a query q. Without any additional knowledge about the user, we propose and study two complementary basic interpretations of matching:

Conjunctive matching: In this interpretation, we assume that a user's notion of "matching" is that document d matches all the words in query q, i.e., $p(Match = 1|q, d) = \prod_{w \in q} p(w|d)$ in order to be retrieved. With this interpretation, the model of recall can be refined as follows,

$$\mathbf{Rec}(q, \mathcal{R}_K) = \prod_{w \in q} p(w|\mathcal{R}_K). \tag{3.7}$$

We note that such a conjunctive interpretation has an inherent bias toward favoring short queries since adding a word to a query would cause the product to be smaller. Intuitively, this bias makes sense since it would be easier to match all the words in a shorter query than in a longer query. However, we want to consider variable lengths of the query when finding an optimal query; thus, we need to normalize the product using the query length. We thus add an exponent $1/|q|$ to the product, where $|q|$ is the total number of words in the query (query length), giving an interpretation of the "per-word" query probability in the conjunctive recall model (Cr).

$$Cr : \mathbf{Rec(q, \mathcal{R}_K)} = (\prod_{w \in q} p(w|\mathcal{R}_K))^{1/|q|}. \tag{3.8}$$

We can similarly refine the model of precision to obtain the following conjunctive precision model (Cp)

$$Cp : \mathbf{Prec(q, \mathcal{R}_K)} = 1 - (\prod_{w \in q} p(w|\bar{\mathcal{R}}_K))^{1/|q|}. \tag{3.9}$$

Disjunctive matching: Alternatively, we can also interpret matching as the document matching any one of the query words, i.e., $p(Match = 1|q, d) = \frac{1}{|q|} \sum_{w \in q} p(w|d)$. Applying this interpretation, we can obtain the following disjunctive recall model (Dr) and disjunctive precision model (Dp)

$$Dr : \mathbf{Rec}(q, \mathcal{R}_K) = \frac{1}{|q|} \sum_{w \in q} p(w|\mathcal{R}_K), \tag{3.10}$$

$$Dp : \mathbf{Prec}(q, \bar{\mathcal{R}}_K) = 1 - \frac{1}{|q|} \sum_{w \in q} .p(w|\bar{\mathcal{R}}_K) \tag{3.11}$$

Note that the disjunctive interpretation is naturally normalized without any length bias and the conjunctive and disjunctive interpretations can be combined, leading to four different instantiations of the PRE framework i.e., *CrCp*, *CrDp*, *DrCp*, *DrDp*. Moreover, we can use a higher-order n-gram language model (e.g., a bigram language model) to replace the unigram language model in either of the two interpretations above to potentially achieve more accurate modeling and generate more variations. Thus PRE can serve as a roadmap for us to explore better models for query formulation by systematically improving each component model.

3.4.2 Instantiation of Effort

We instantiate the effort function as a constraint such that only queries of length less than a threshold L would be allowed, i.e., $\mathbf{E}(q)$: $|q| <= L$. As $\mathbf{E}(q)$ is a constraint, it is equivalent to taking $\mathbf{E}(q)$ as an indicator function and setting λ parameter (weight of $\mathbf{E}(q)$ in Eq. 3.4) to a sufficiently large constant such that the length boundary L would in effect act as the weight parameter influencing the importance of effort.

3.4.3 Solving optimization problem

Even with a restricted length, the complexity in solving the optimization problem is still exponential as the query can contain any words in any order. To address this problem, following previous works [14, 43, 95], we use a modified greedy algorithm to first generate candidate queries and then find the optimal query. Using the vocabulary of words in the prototype document $\mathcal{R}_K$, we first enumerate all the one-word and two-word queries as an initial set of candidate queries. The two-word candidate queries are then expanded greedily by adding a word that maximizes the whole query score, creating increasingly longer queries until L-word candidate queries are created. All the candidate queries (with variable lengths) are finally ranked according to their optimization function score resulting in a ranked list of candidate queries. The top query is taken as the simulated query. During reformulation, the previously simulated queries are ignored from the candidate query list so as not to duplicate previous queries.

3.4.4 Knowledge state update

As described in Section 3.3.3, a user's knowledge K is $\{\mathcal{R}_K, \bar{\mathcal{R}}_K\}$. We assume that the initial knowledge of the user regarding relevant and non-relevant information is based on information need description (s) and collection ($\mathcal{C}$), respectively. That is, $\mathcal{R}_K = s$ (s is the only relevant document) and $\bar{\mathcal{R}}_K = C$ (all documents in the collection are non-relevant and can be concatenated into one single prototype non-relevant document). This assumption is appropriate for building (dynamic) user simulators based on static TREC collections [14].

During a search session, the user can scan through the search results to learn new information. In this process, the user would be exposed to examples of both relevant and non-relevant documents, which can then be used to update $\mathcal{R}_K$ and $\bar{\mathcal{R}}_K$, respectively, to enrich the representation of relevant and non-relevant information.

With the unigram language model instantiations the simulated user would only need to store the knowledge about relevance in the form of two unigram LMs, i.e., the relevance LM $p(w|\mathcal{R}_K)$ and the non-relevance LM $p(w|\bar{\mathcal{R}}_K)$, which model the probability that word w

occurs in a prototype relevant document and non-relevant document, respectively. With such a knowledge storage model, updating of knowledge boils down to updating these two LMs by aggregating word counts from the newly acquired examples of both relevant and non-relevant documents as follows

$$p(w|\mathcal{R}_K) = \frac{c(w,s) + \sum_{sr} c(w,sr)p(R=1|sr,s)}{\sum_{w'} c(w',s) + \sum_{sr} c(w',sr)p(R=1|sr,s)}, \tag{3.12}$$

$$p(w|\bar{\mathcal{R}}_K) = \frac{c(w,\mathcal{C}) + \sum_{sr} c(w,sr)p(R=0|sr,s)}{\sum_{w'} c(w',\mathcal{C}) + \sum_{sr} c(w',sr)p(R=0|sr,s)}, \tag{3.13}$$

where $c(w,s)$ gives count of the word w in s, $c(w,\mathcal{C})$ is the count of the w in the collection $\mathcal{C}$, sr is a snippet of a retrieval result, and $p(R=1|sr,s)$ and $p(R=0|sr,s)$ are the probability that snippet sr is relevant or non-relevant, respectively, estimated based on known relevance judgments of the corresponding documents. Once the relevance and non-relevance LMs are updated, they can be used to reformulate a query using the PRE framework. As in the case of refining the models of recall and precision, more sophisticated language models and knowledge updating mechanisms can be easily plugged into the PRE framework.

3.5 EXPERIMENT DESIGN

The purpose of our experiments is to study the soundness and benefit of the proposed framework and answer the following research questions: **RQ1:** Do the empirical results support the design of the objective function of PRE, i.e., maximization of both precision and recall? **RQ2:** Do the improvement of the precision and recall instantiations have an additive effect on the overall performance improvement, or is there a complex interaction between precision and recall models? **RQ3:** Does the relative performance of different query simulation methods follow a similar trend for both initial and reformulated queries? In the rest of this section, we describe how we design our experiments to answer these questions.
Dataset: We use TREC Session Track 2012, 2013 and 2014 data sets [110] because they are among the very few data sets with the initial and reformulated queries formulated by real users, which we need for evaluation. Each topic is an information need with a title and description. Each topic's description is used as an information need description s to perform query simulation. We used unigram word frequencies derived from Google Web Trillion Word Corpus [115] available for the top 333,333 words as the collection language model $p(w|C)$. We used indri and pyndri [116] for indexing the ClueWeb datasets. For evaluating the simulated queries, we compared them with the real user queries of the same information need obtained from the sessions in the Session track (St for short) datasets. St 2012 and 2013 only have

200 and 400 user queries respectively, compared with 3600 user queries in St 2014. Thus we can expect that the evaluation with St 2014 is more consistent and robust. Further, St 2012 has completely different topics compared to St 2013, 2014 whereas St 2013 and St 2014 share some topics.

Query similarity measurement: To assess the quality of a simulated query, for every simulated query, we computed its maximum Jaccard similarity [117] with any of the real user queries of the same information need. We use Jaccard similarity because the TREC topic descriptions used by the real users and our simulation model are the same, so it is likely that the users have used most of the words from this topic description. The average of these similarities for all simulated queries of all TREC topics in the dataset is computed as average Jaccard similarity (Avg_jsim). Avg_jsim can be computed with *top-k* simulated queries from the ranked list generated by PRE. The reason to compute maximum similarity is that the simulated query can be similar to any one of the real user queries and need not be close to all.

$$Avg\_jsim@k = \frac{1}{|T|}\sum_{t\in T}\frac{\sum_{Q\in smt_{t,k}} \max_{q\in act_t}(jsim(Q,q))}{|smt_{t,k}|} \tag{3.14}$$

where $jsim$ is Jaccard similarity, act_t is the set of real user queries, $smt_{t,k}$ is the set of top-k queries generated for that topic and T is the set of all topics (information needs).

In addition to Jaccard similarly, we also compute F-measure score of the simulated queries. For each simulated query, we compute how many words of the real query are covered (recall) and how many words in simulated query are actually in the real query (precision). We use average recall and average precision to compute the final F-measure score. Similar to Jaccard similarity, we compute maximum recall and maximum precision score for a simulated query when compared to a real user query.

$$\begin{aligned} Avg\_prec@k &= \frac{1}{|T|}\sum_{t\in T}\frac{\sum_{Q\in smt_{t,k}} \max_{q\in act_t}(prec(Q,q))}{|smt_{t,k}|} \\ Avg\_recall@k &= \frac{1}{|T|}\sum_{t\in T}\frac{\sum_{Q\in smt_{t,k}} \max_{q\in act_t}(recall(Q,q))}{|smt_{t,k}|} \end{aligned} \tag{3.15}$$

where $prec(Q,q) = \frac{|Q\cap q|}{|Q|}$, $recall(Q,q) = \frac{|Q\cap q|}{|q|}$.

Parameter estimation and setting: We estimate parameter α in Eq. 3.4 using four fold cross-validation; it can vary from 0 to 1 and we used grid search with 0.1 step. We set the effort parameter, which is the threshold of the effort function or the maximum query length L to 6. We have also studied the effect of varying the parameters α and L on the performance results.

Table 3.1: $Avg\_jsim@5$ for Rec, Prec combination methods using $L = 6$ for Session Track 2012-2014 data.

Jaccard similarity	**Session track datasets**		
Methods	**St 2012**	**St 2013**	**St 2014**
CrCp	0.2536 (0.05)	0.3612 (0.475)	0.4431 (0.125)
CrDp	0.267 (0.1)	0.3539 (0.1)	0.436 (0.1)
DrCp	**0.253 (0.125)***	**0.3651 (0.85)**	**0.4504 (0.7)**$^{*,!}$
DrDp	**0.273 (0.1)**	0.3533 (0.1)	0.4436 (0.1)
Cr	**0.2529**	**0.3435**	**0.4124**
Dr	0.2407	0.3312	0.4045
Dp	0.2038	0.1453	0.313
Cp	0.2538	0.2108	0.3573
QS3+ (baseline)	0.2696	0.2368	0.4049

To establish statistical significance between the performance of two different methods, we perform an independent two-sample t-test with a p-value of 0.05; the p-value is chosen as 0.05 as there are very few instances, just the number of TREC topics, of Avg_jsim.

3.6 EXPERIMENT RESULTS

We first compare different instantiations of Rec and Prec and the four combinations of Rec and Prec instantiations, which we refer to as combination methods, in Table 6.2 and Table 3.2. Table 6.2 and Table 3.2 show the average Jaccard similarity and F-measure scores of all the instantiations. We can infer from Table 6.2, that combination methods are always better than individual Rec or Prec methods (which are special cases when we set α to either 1 or 0), i.e., $CrCp$, $CrDp$, $DrCp$, $DrDp$ always perform better than Cr, Dp, Cp, Dp in all datasets. Similar trends can also be observed in Table 3.2. In Table 6.2 and 3.2, ! indicates statistically significant difference between $DrCp$ and Cr which are best methods among combination methods and individual component methods respectively. This shows that both Rec and Prec are important in the query formulation process; using recall or precision alone always performs less than combining them together. The result provides some empirical justification for the design of the objective function of the PRE framework to optimize both precision and recall, thereby answering **RQ1**.

Among individual components, Cr always performs better than Dr and Cp is performing better than Dp. However, among the combination methods, we observe that combining Dr with Prec methods performs better than combining Cr with Prec methods. $DrCp$ performs better than $CrCp$ and $DrDp$ also performs better $CrDp$. But in the case of precision, Cp mostly performs better than Dp even after combining with a recall model. This implies that

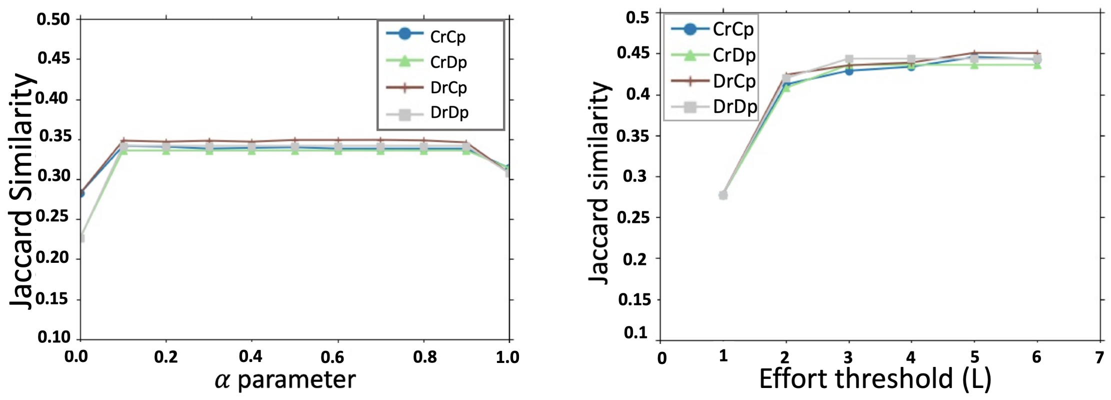
Table 3.2: $F - measure@5$ for Rec, Prec combination methods using L=6 for Session Track 2012-2014 data.

F-measure	**Session track datasets**		
Methods	**St 2012**	**St 2013**	**St 2014**
CrCp	0.3893 (0.05)	0.5324 (0.275)	0.6373 (0.15)
CrDp	0.3991 (0.1)	0.5169 (0.1)	0.6081 (0.1)
DrCp	0.3876 (0.125)	**0.5344 (0.875)***	**0.6474 (0.625)**$^{*,!}$
DrDp	**0.4071 (0.1)**	0.5167 (0.1)	0.616 (0.1)
Cr	**0.3855**	**0.4958**	**0.5895**
Dr	0.3663	0.4822	0.5854
Dp	0.3161	0.2436	0.4702
Cp	0.3836	0.3459	**0.5569**
QS3+	0.3886	0.3726	0.5664

combining the best individual methods in Rec and Prec may not always result in the best method overall, indicating that there may be a complex interaction between the Rec and Prec models, which has to be further studied to optimize the component models of PRE (addressing **RQ2**).

The disjunctive and conjunctive matching is not influenced by the length of a query when they are length normalized; therefore, the longer queries or short queries are scored based on their aggregated matching to relevant and irrelevant documents. The conjunctive matching methods Cr and Cp are proportional to the geometric mean of the query words matching scores, and the disjunctive matching methods Dr and Dp are proportional to the arithmetic mean of the matching scores. Thus Cr is more strict and requires all the query words to have better matching scores for a higher Cr, whereas Dr can be a higher value even if some of the query words have high matching scores. Similarly with Cp and Dp.

As individual components, one of the reasons that Cr performs better than Dr may be that conjunctive matching requires all words to be well matched by the document and thus is more strict than disjunctive matching; therefore it helps retrieve only very relevant documents, so it indirectly avoids irrelevant documents. However, when combined with the precision component, which already minimizes the irrelevant documents itself, the disjunctive matching for recall Dr performs better than Cr because potentially, the disjunctive matching allows for more documents to be matched as it can result in a higher score with matching any of the query words and does not require perfectly matching all words of the query. Thus, Dr is better for maximizing recall alongside any precision component and therefore results in a better query. Similarly, Cp is better than Dp because Cp avoids irrelevant documents only if all query words match the document, whereas Dp avoids a document even if some of the



(a) varying α (b) varying effort threshold (L)

Figure 3.1: $Avg\_jsim@5$ with varying α and L

query words match the irrelevant information, which makes Dp more strict and potentially miss relevant documents; which may be the reason for Cp performing better than Dp among the precision components.

As a reference point, we also include the performance of a state-of-the-art baseline model QS3+ [39] in Table6.2 and Table3.2. We see that even the very basic instantiations we have studied in this chapter can already outperform this baseline method, $*$ indicates a statistically significant difference between $DrCp$ and $QS3+$ in Table 6.2 and 3.2, suggesting great potential for using PRE to further develop more effective query simulation methods. Among all the PRE instantiations, $DrCp$, the disjunctive instantiation of recall (Dr) combined with conjunctive instantiation of precision (Cp) appears to be most effective for query simulation except in St 2012 dataset. However, as these are only basic instantiations of PRE, we expect to be able to achieve better performance in the future as we further explore more sophisticated ways to instantiate PRE (e.g., higher-order n-gram LMs).

Table 6.2 and Table 3.2 also show the optimal parameter value of α obtained through cross-validation in the parenthesis for each method. For Cr, Cp, $\alpha = 1.0$ and for Dr, Dp $\alpha = 0.0$ as they are recall only and precision only methods respectively. For the remaining methods, we observe that the optimal α is mostly 0.1, with the exception of $DrCp$, which has $\alpha = 0.85$ and $\alpha = 0.7$ as the best parameter. Overall, the optimal parameter is never 0.0 or 1.0, which again shows that considering both recall and precision gives the best performance for the instantiations. To address **RQ3**, we now perform query reformulation using search results of the initial query with the knowledge state update method described in Section 3.4 to see if some conclusions we have made on initial query formulation also hold for reformulated queries. In Table 3.3, $Avg\_jsim@5$ is computed by comparing the reformulated queries with actual queries. We observe similar trends in performance in Table 6.2 and

Table 3.3: *Avg_jsim*@5 of reformulated queries

Methods	**St 2012**	**St 2013**	**St 2014**
CrCp	0.2520	0.3639	0.4333
CrDp	0.2606	0.3575	0.4264
DrCp	0.2523	**0.3743**	**0.4383**
DrDp	**0.2670**	0.3616	0.4360

Table 3.3. The reformulation method further ensures that *DrCp* is the best method in most cases, and similarly *DrDp* performs best on St 2012. Similar trends are also observed for the F-measure score. These results suggest that the uniform modeling of initial formulation and subsequent reformulation is reasonable and also facilitates optimization of a query simulator by optimizing knowledge state updating and query generation separately.

One of the reasons why St 2012 results are consistently different from other datasets could be because of the small size of the dataset and the different topic set compared to the remaining datasets. The different trends of the results for different datasets shows that may be single instantiations is not optimal for all datasets and different instantiations of the PRE framework may be better for different set of information needs and document collections. This is an interesting result because it further shows that arbitrary methods for query simulation may not be best for all datasets and the empirically validation is only limited to those datsets utilized. The PRE framework is a general framework and the experiments show that the soundness of the optimization framework holds for different datasets even if the best specific instantiation varies.

3.6.1 Analysis of parameters

We analyzed the sensitivity of all the combination methods with respect to parameter α as shown in Figure 3.1a. Figure 3.1a shows *Avg_jsim*@5 for different α's over all the folds in cross-validation. The performance is low for $\alpha = 0.0$ and 1.0, and mostly the same for α between 0.1 to 0.9, and a similar trend is observed for both Jaccard similarity and F-measure. Thus, it can be concluded that as long as both recall and precision are considered in the optimization framework, the performance of different instantiations is not highly affected by α. These results also indicate that the PRE instantiations consistently outperform baseline model QS3+ irrespective of the parameter α as long as both recall and precision are considered.

We have studied the distribution of best α for different users in the St 2014 dataset as each session is associated with a *user id*. For each user, we have estimated the best α that optimizes *Avg_jsim*@5 of the simulated queries by grouping sessions of that user. We could obtain two interesting observations from these results. First, the performance variation by varying α for each user is similar to that of the average variation in Figure 3.1a, further supporting our conclusion that considering both recall and precision is necessary. That is, $\alpha = 0.0$ and 1.0 are non-optimal not only for the overall average performance with all sessions but also for each individual user's sessions. Second, we examined the best α for those users where the performance is most sensitive to α, and found that the best α varies to a great extent between each user, which indicates that real user behavior varies with different tradeoffs between recall and precision; thus, not only should we maximize both recall and precision, but we also must have the parameter α in PRE in order to accurately simulate individual users' variable tradeoff between precision and recall.

As described in Section 3.5, we have set maximum length threshold (L) to *6*, $\mathbf{E}(q) : |q| < 6$, for obtaining results in Table 6.2, 3.2, 3.3. To study the sensitivity of this parameter, we have varied the threshold L from 1 to 6 and the performance of different combination methods are shown in Figure 3.1b, we observe that the performance of all the methods increases in the beginning from $L = 1$ to 3 and then is mostly the same for $L = 3$ to 6. Indeed, when the threshold is 6, we observed that most of the optimal queries are of length 2 or 3 and the average query length of the simulated queries is 1.97 for *St 2014*. This shows that the PRE framework cuts down the query length automatically for the optimal query even with higher threshold parameter. At the same time, it can also simulate long queries (even of length five) as shown by the example simulated queries in Table 3.4. Therefore, unlike many existing simulation approaches which choose a constant query length, PRE adaptively simulates long or short queries depending on the information need and setting L to 6 will allow for simulating a longer query whenever it is optimal.

3.6.2 Qualitative analysis

In Table 3.4, we provide a few examples of the simulated queries from top-10 queries generated by the *DrCp* method and similar real user queries for those topics. From the results, it can be observed that the quality of the queries highly depends on the information need description as it is the only information used. If the important words representing the information need are frequent in the description, then simulated queries are close to the user queries and vice versa; for example, as "swahili dish", "hydropower" appear many times in the topic description so the simulated queries are close to user queries for the first and third topics in Table 3.4, whereas the important keyword "world cup" appears only once in the last example topic which is probably the reason for poor quality simulated queries, thus this also leads to poor performance sometimes. The results also show that the query length can either

Table 3.4: Examples of simulated queries

Topic description	Similar real user queries	Simulated queries
A friend from Kenya.......surprise him .... cooking a traditional Swahili dish..learn about Swahili dishes and how to cook them. Find..about Swahili home cooking.	swahili food traditional, swahili dish wiki, swahili traditional dish, swahili dish, swahili cook.	swahili, swahili dish, dish, traditional swahili dish.
You are planning a winter vacation to the Pocono Mountains region...Where will you stay? What will you do while there? How will you get there?	pocono mountain winter activity, winter vacation in the pocono mountain, pocono mountain region.	winter pocono mountain vacation plan, pocono mountain region winter vacation plan, pocono pennsylvania winter vacation plan.
Hydropower...renewable sources of energy..replace fossil fuels. Find information about the efficiency of hydropower, the technology..consequences building hydroelectric dams.. on the environment.	hydropower efficiency, what be hydropower, hydropower damn, hydoelectric power, hydropower environment	hydropower, hydropower hydro-electric, hydropower dam, hydropower efficiency.
France won..World Cup in 1998. Find information about the reaction of French people and institutions (such as stock markets), and studies about these reactions.	1998 world cup french reaction, french world cup 1998, french reaction win world cup, stock market world cup 1998.	reaction, 1998 reaction, cup reaction, reaction institution, france reaction.

be long or short depending on the information need; for example, in the second example of Table 3.4 the queries generated are very long, similar to real user queries which are also long.

3.7 DISCUSSION AND FUTURE WORK

Ideally, user simulation should be done based on a computational user model that can explain how users make various decisions (e.g., querying or viewing documents) in the search process and suggest an operational algorithm that can generate user actions in response to a search engine. However, we are currently far from such a model. For example, none of the current methods for query simulation suggests any hypothesis about the process that a user might have followed in formulating a query. This is partly due to the lack of understanding of the internal cognitive and reasoning processes in a user's mind, which requires more progress in research in cognitive science, information science, and human-computer interaction.

We can approach such an ideal computational model of users in two directions: 1) We can do user studies to sufficiently understand search users to be able to design a computational user model to simulate the understood user behavior. 2) We can design a computational user model that is as explanatory as possible of the user behavior. The proposed PRE framework can be regarded as taking the second direction and making a step toward building an interpretable and explanatory computational model for simulating a user's query formulation/reformulation process. PRE is designed based on multiple hypotheses about how exactly a user formulates a query. Our preliminary evaluation results provide some evidence to support the hypotheses, but clearly, more research is needed to further assess the validity of the hypotheses proposed in PRE, especially via designing appropriate user studies or leveraging search log data to more rigorously examine those hypotheses.

PRE is an interpretable framework. The query generated by the model can be explained in terms of which objective is optimized by the query and to what extent, and the knowledge state that leads to the query. The parameters of the model controlling the weight of different components can be varied in order to simulate different user strategies while querying. The knowledge state can also be further varied from high to low knowledge state by changing the vocabulary size and probabilities to mimic users with an expert or poor vocabulary knowledge while formulating queries.

As a novel framework, PRE opens up many interesting opportunities for future research.

First, PRE can easily accommodate the incorporation of or combination with additional formal models of user behavior to further advance the development of interpretable computational models of users. For example, any click models or browsing behavior models can be combined with PRE to provide a more complete simulation model of a user.

Second, from a practical viewpoint, PRE offers three lines of immediate benefits: 1) It provides a general formal framework that we can use to compare the existing query simulation methods and systematically examine their effectiveness from the perspective of optimizing precision and recall and minimization of effort. 2) It facilitates the design of many new simulation models via different ways to instantiate each component of the PRE framework. This is a promising direction as even the very basic instantiation strategies explored in this chapter already deliver comparable performance to one of the frequently used state of the art query simulation method *QS3+* [39]. For example, *DrCp* instantiation method outperforms *QS3+* for Session track 2013 and 2014 datasets. There is much potential to further improve the effectiveness of query simulation using the PRE framework by improving instantiations of the objective models of PRE. 3) Its interpretable parameters enable simulation of meaningful user variations in multiple dimensions, including, e.g., variation in the relative importance of precision and recall, variation in user's initial knowledge state, variation in effectiveness of a user's knowledge state update during search session, and variation in the tradeoff between the effort and effectiveness of the formulated query. Exploration of such directions is an important future work.

Third, PRE provides a theoretical roadmap for further exploration of new ways to potentially model how a user estimates precision and recall more accurately. We discuss two specific possibilities here. 1) As a more intuitive way to factor out precision and recall, the models of precision and recall can be defined more similarly to the precision and recall measures on retrieval results as follows,

Expected_precision (q) $= \frac{\sum_{d \in C}(p(Match=1|q,d)p(rel|d)}{\sum_{d \in C} p(Match=1|q,d)}$,
Expected_recall (q) $= \frac{\sum_{d \in C}(p(Match=1|q,d)p(rel|d)}{\sum_{d \in C} p(rel|d)}$,
where $p(rel|d)$ is the probability that the document d is relevant. 2) The models of recall and precision can also be computed over subtopics or facets of relevant information (like in [94]) instead of the set of relevant documents. The set of relevant documents of an information need often consists of multiple subtopics or facets and to satisfy the information need, the user may want to care more about maximizing recall and precision of the *subtopics* than those of the documents in the retrieved results. Such a new model has the potential for modeling query formulation in the context of complex retrieval tasks.

Finally, the conclusions we have drawn in our experiments can be further examined by performing more experiments and using additional evaluation measures. Our use of Jaccard similarity is justified based on the fact that the simulator model and most of the real users used the same word set from the topic descriptions, but it does not support inexact matching of semantically related words. Although whether to support inexact matching appears to be orthogonal to the hypotheses we are testing, meaning that adding inexact matching will unlikely have a significant impact on the conclusions we have drawn, it is necessary to further experiment with other evaluation measures [36, 89, 90] including new semantic similarity measures such as BLEURT [118] in the future to further verify our conclusions.

In the instantiations of the PRE model, we have not explored the information need variation and different types of knowledge components and how they affect the query formulation. For example, the user can explore more related to their information need or only focus on getting the required information, and it is important to be able to see such variation in user behavior while simulating search sessions. In our next chapter, we propose a cognitive state user model to model the information need and knowledge state of a user in detail and propose parameters associated with different behaviors of users. The cognitive state model is used for query reformulation and click model in the context of e-commerce search.

Additionally, we have used a simplified language model in PRE to represent the knowledge state (knowledge of relevant and irrelevant terms) and, therefore, to compute the recall and precision components. But if we have access to the user interaction data like user search logs, we can use that to further learn or train better models to compute recall and precision and cost of making the query. In general, user search log data contains rich information about user interaction which can be leveraged by the user simulation models to simulate a wide range of user search intents and behavior. In the next chapters, we propose user simulation models leveraging user search logs, where one of them is an unsupervised method CSUM model, and the other is a supervised imitation learning model.

Chapter 4: A COGNITIVE USER MODEL FOR E-COMMERCE SEARCH

In this chapter, we address the question of **how to model a user's cognitive state in user simulation**. We present a Cognitive state user model (CSUM) which is an interpretable generable model that simulates user actions in a search session based on a dynamic user cognitive state consisting of user knowledge and information need. We chose to study this problem in the domain of E-com primarily because of the availability of the search logs in this domain that we had access to. Additionally, the search intent of the user is relatively structured in the E-commerce domain in terms of preference over product domain attributes. However, the ideas can be generalized to other search domains as well. The proposed user model can be used both for mining E-commerce search logs and also for simulating user actions, we evaluate the model for both applications.

4.1 INTRODUCTION

We present a novel cognitive user model for E-Commerce (E-Com) search that goes beyond existing user models to model a user's cognitive state, including the information need and knowledge state of the user. The model includes components to capture all the major search actions of a user, including query formulation, clicking on results, and query reformulation, and thus provides a complete model for users of E-Com search. The model has interpretable parameters that can be estimated using E-Com search log data to analyze and understand users' behavior as well as can be manually adjusted to simulate different kinds of users.

Our proposed model CSUM is a conditional generative user model for generating user actions that can be used for simulation and for mining E-Comm search logs to understand users' behavior. The main hypothesis of the proposed model is that the generation of all important user actions in the whole user session is conditioned on the target product that the user attempts to buy (information need) and a starting point like the initial query. Compared with previous work, the novelty of our model is that: 1)it explicitly models the user cognitive state and updates the state to model different user behavior characteristics in one unified model, 2)the query formulation/reformulation model is studied in-depth and the query reformulations in the search log are also used for analysis, 3)the interpretable model parameters can be learned from the search log and it can be learnt from even a single session to identify specific user behavior. Our model has several benefits, it can be used (a) as a tool to identify interesting user behaviour patterns in search logs, (b) to simulate user sessions given the input formulation like the target product, and starting query. We evaluate the model on an E-comm search log and show that the model can be used to analyze user behavior in multiple ways. Our results on an E-Comm search log show that the model performs well in predicting user actions in a search session. The analysis of search logs reveals multiple interesting findings about the E-com search behavior like (1) users generally differ most in exploratory/non-exploratory search behaviour (2) the variance in behaviour is larger for the same user while purchasing different products than compared to that of different users buying the same product. Finally, the CSUM model can be applied to simulate search sessions with varying user behaviours by altering the parameters.

4.2 A COGNITIVE STATE USER MODEL (CSUM)

CSUM is a conditional generative model conditioned on some assumed Information need (IN) and knowledge that can be based on the input. Given such an assumed/initial IN and knowledge, the model would then attempt to model (1) how a user formulates the very first query, (2) how a user might reformulate the query (as needed), and (3) how a user would respond to the search results (which result to click on) (4) how the IN and knowledge will be updated as the user interacts with the search system which can be potentially repeated multiple times. In CSUM, the user cognitive state consists of IN and knowledge. We now describe CSUM in more detail. Different components of the proposed model CSUM and how they are connected to interact with a search engine are shown in Figure 1.

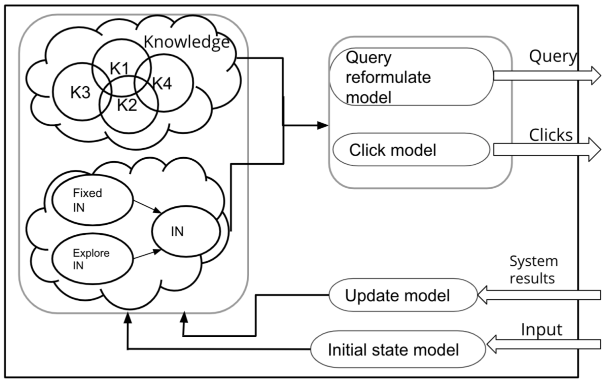


Figure 4.1: Outline of the proposed CSUM model

4.2.1 Representation of IN

We represent the IN as a set of probability distributions of all those attribute values of the products preferred by a user.

One simple way to define IN is to fix it to a target product the user may purchase in the end(i.e., setting all the attribute value probabilities to 1.0), which we will refer to as a ***Fixed IN*** model. However, *Fixed IN* is only appropriate (accurate) if the user indeed is specifically interested to purchase a particular product and had never changed preferences during the search process but there is a possibility for uncertainty in preferences. To model this uncertainty, we further introduce an ***Exploring IN*** model where the probability of the target product's attribute value is still relatively large, but not 1.0, thus leaving room for exploration.

Let the probability distributions for *Fixed IN* and ***Exploring IN*** be denoted as $P(V_{A,a_i}|fixed\_IN)$, $P(V_{A,a_i}|explore\_IN)$ where A is the attribute which can be category (C), brand (B), and title words (Ti), and a_i are the attribute values of A.

We assume that at any time, the user's information need may be "between" the *Fixed IN* and *Exploring IN*, and this preference is potentially attribute-specific. The uncertainty is captured by a mixture model with $P(f_A)$ indicating the probability of the user being in the state of *Fixed IN*, thus 1-$P(f_A)$ is the probability of being in a state of *Exploring IN*.

The user IN is computed as follows:

$$P(f_A) * P(V_{A,a_i}|fixed\_IN) + (1 - P(f_A)) * P(V_{A,a_i}|explore\_IN) \tag{4.1}$$

4.2.2 Representation of Knowledge State

From a cognitive IR perspective, the user may have knowledge acquired from different types of sources. We model the knowledge state by four components as follows, 1)**Background Knowledge(K1)**: It represents the background knowledge of the user about how a search system understands the user query. A user may have prior knowledge of these mappings based on previous experience or general knowledge of the E-commerce product space. It is denoted by $P_{kb}(cat/w_q)$, $P_{kb}(w_p/w_q)$ which is the likelihood that a query word w_q would lead a search engine to return a particular category of products *cat* or return a product with word w_p occurring in the brand field or the title field. 1)**Knowledge Learnt(K2)**: The user can learn about the product space during the search session using the search results obtained for a query. It is denoted by $P_{kl}(cat/w_q)$, $P_{kl}(w_p/w_q)$. 1)**Keyword matching knowledge(K3)**: It is the knowledge of the product space vocabulary that is the words present in the text description of different attributes as the search system also uses direct keyword matching $P(w|wa) = 1$ if $w = wa$, 0 otherwise, wa is set of all words in target product. 1)**Linguistic knowledge (K4)**: In addition to knowledge of product space, the user further has linguistic knowledge of word meanings. We model this knowledge by normalizing the word similarities obtained through word embeddings to make the probability distribution $P_{sim}(w_1|w_2)$.

4.2.3 Update model

The IN and knowledge are updated in the following way using the *Update model* to simulate how the user cognitive state is updated during a session. We update the IN and knowledge each time after interacting with a page of results from the system, i.e, after viewing and clicking products on one result page. The next action can be moving to the next page or reformulating the query or purchasing or stopping the session.

Information need update

As we know the final purchased product is given by *Fixed IN*, thus the uncertainty progressively reduces towards it and therefore $P(f_A)$ or P(*Fixed IN*) is increased during a session. The amount of increase is modelled through the following heuristics.

1. If the user observes more attribute values from the target product or *Fixed IN* in the results, $P(f_A)$ increases proportionally. A parameter λ_1 controls the weight of this factor. $P_o(f_A) \propto \lambda_1 * P(V_{A,\{o_i\}}|fixed\_IN)$ where $\{o_i\}$ is the set of observed attribute values among the results.

2. As the session gets longer, the user tries to decide thus $P(f_A)$ increases based on session length, λ_2 is the parameter here. $P_{session}(f_A) = \frac{l}{f(\lambda_2)+l}$ where $f(\lambda_2)$ is linear function of λ_2, l is the length of the session until then.

3. If a user clicks a product, the preferences $P(V_{A,a_i})$ for the a_i of the product are updated. If the a_i of the clicked item is different from the final purchased product, it is likely user dislikes the product, thus $P(V_{A,a_i}) = P(V_{A,a_i}) * (1 - \beta_{iupdate}) \forall a_i \in Clk_p \setminus T$ where T indicates the target product , Clk_p indicates the clicked product.

Finally, the $P(f_A)$ is updated as follows,

$$\begin{aligned} P_{update}(f_A) &= P(f_A) + P_o(f_A) + P_{session}(f_A) \\ P(f_A) &= \frac{P_{update}(f_A)}{P_{update}(f_A) + (1 - P(f_A))} \end{aligned} \tag{4.2}$$

Knowledge Update

Similar to IN, the user learns and obtains more knowledge of the product space while interacting with the system. We only update *Knowledge learnt* (K_2) component of knowledge, leaving the exploration of a more sophisticated update model for knowledge as future work.

The probability $P_{kl}(cat|w_q)$ is directly proportional to the count of the number of products from the result page that belong to category *cat* and have the word w_q among its attribute values. Similarly, the probability $P_{kl}(w_p|w_q)$ is proportional to the number of products from the result page which have both w_p and w_q among any of its attribute values. Thus, P_{kl} is iteratively updated with the new knowledge learnt from that set of result items.

Formally, let $f_1(r_p, clk_p)$, $f_2(w_p, r_p, clk_p)$ be the functions of a result product p which is ranked at position r_p and clk_p is a boolean value which denotes whether the product is clicked or not. $f_1(r_p, clk_p)$, $f_2(w_p, r_p, clk_p)$ are used to compute $P(cat|w_q)$, $P(w_p|w_q)$ respectively. According to our hypothesis, the functions f_1, and f_2 are inversely proportional to r_p and directly proportional to clk_p, and f_2 is directly proportional to the frequency of w_p in the product. Let w_p denote all the words in the attribute values of product p. Therefore,

$$P_{kl}(cat/w_q) = \frac{\sum_{p \in \{S_{w_q,cat}\}} f_1(r_p, clk_p)}{\sum_{cat'} \sum_{p \in \{P_{w_q,cat'}\}} f_1(r_p, clk_p)} \tag{4.3}$$

$$P_{kl}(w_p/w_q) = \frac{\sum_{p \in \{S_{w_q,w_p}\}} f_2(w_p, r_p, clk_p)}{\sum_{w_{p'} \in W_S} \sum_{p \in \{S_{w_q,w_{p'}}\}} f_2(w'_p, r_p, clk_p)} \tag{4.4}$$

$\forall w_q \in query\,words$, where the $\{S_{w_q,cat}\}$ is set of result products which have w_q among the words of their attribute values and cat as the category. Similarly $\{S_{w_q,w_p}\}$ is set of results products which consist of w_q and w_p among the words of their attribute values. W_S are all words in product titles and brand attributes of all result products $\{S\}$.

If the session length is more than one, then for each result page the $P_{\{kl,new\}}(cat/w_q)$, $P_{\{kl,new\}}(w_p/w_q)$ are computed in the same way as above and are updated to the previous *Knowledge learnt* $(P_{kl}(cat/w_q), P_{kl}(w_p/w_q))$. The parameter $\beta_{kupdate}$ is used to update the *Knowledge learnt* during the session.

$$\begin{aligned} P_{kl}(cat/w_q) &= P_{kl}(cat/w_q) + \beta_{kupdate} * P_{\{kl,new\}}(cat/w_q) \\ P_{kl}(w_p/w_q) &= P_{kl}(w_p/w_q) + \beta_{kupdate} * P_{\{kl,new\}}(w_p/w_q) \end{aligned} \tag{4.5}$$

4.2.4 Query Reformulation model

Query reformulation strategy is modelled as removing and adding query words (only adding words for formulating the initial query). This is done by scoring a query as a function of individual word scores and the user tries to maximize the query score. To remove or add a word, the query score is optimized by removing or adding words that increase the query score compared to the current query.

In the following, we describe the word scoring method. The score of a word depends on different knowledge sources the user might have like *Background knowledge (K_1)*, *Knowledge learnt (K_2)*, *Keyword matching knowledge (K_3)*, *linguistic Knowledge (K_4)* and the *Informa-*

tion need (IN). Each knowledge source is given a weight indicating the importance of that source for scoring words. The score for a potential query word can be understood as the combination of score based on Information need(IN) and scores from different Knowledge sources $\{K_i\}$ and weight of the knowledge sources $\{\alpha_{K_i}\}$.

Therefore the query word score is computed as follows,

$$w\_score_A(w) = \sum_{K_i} \alpha_{K_i} * (\sum_{v_{A,a_i}} K\_score(K_i, w, v_{A,a_i}) * IN\_score(v_{Aa}, IN)) \tag{4.6}$$

where K_i is a knowledge source, v_{A,a_i} is either an attribute value or word in the attribute value for attribute A and IN is the current IN of the user's state, α_{K_i} is the weight of the knowledge source K_i. Note that the word score *w_score* is computed for each attribute A (category, brand and title word attributes). The $IN\_score(v_{A,a_i}, IN)$ is given by the probability distribution $P(V_{A,a_i}|IN)$ where a_i is an attribute value of A. The $K\_score(K_i, w, v_{A,a_i})$ is computed from K_i.

Finally, the query score is the average of all the word scores aggregated overall attributes.

$$Q\_score(Q) = \sum_{w_q \in Q} \sum_{A} w\_score_A(w_q) \tag{4.7}$$

To solve for the new query during query reformulation, we solve for words that give the maximum *Q_score* compared to *Q_score* of the current query. Therefore final score of word w is,

$$final\_score(w|IN, \{K_i\}) = Q\_score(Q) - Q\_score(edit\_query) \tag{4.8}$$

where *edit_query* is either $Q - w$ or $Q + w$ referring to removing or adding word respectively. Therefore, a ranked list of words is produced separately for removing words and adding words to the query (only adding words in the case of the first query) and the top-ranked words are chosen according to a numerical threshold or top-k threshold.

4.2.5 Click model

For click decisions, we only use the user IN variable. Based on the IN, relevance and non-relevance models are made. The odds of the likelihood of the product by the relevance model to the non-relevance model gives the probability of clicking it. As the IN is updated, the relevance and non-relevance models are updated accordingly. Therefore the language models are computed as follows,

$$
\begin{aligned}
P(V_{A,a_i}|rel\_model) &= \frac{P(V_{A,a_i}|IN)}{\sum_A \sum_{a_i} P(V_{A,a_i}|IN)} \forall a_i \in A \forall A \in C, B, Ti \\
P(V_{A,a_i}|nonrel\_model) &= \frac{(1 - P(V_{A,a_i}|IN))}{\sum_A \sum_{a_i} (1 - P(V_{A,a_i}|IN))} \forall a_i \in A \forall A \in C, B, Ti
\end{aligned}
\tag{4.9}
$$

Based on a threshold (0.05 is used), the products with a total likelihood score greater than the threshold are clicked in CSUM.

4.2.6 Interpretation of model parameters

The model-specific parameters can be used to understand the user behavior in a session or group of sessions. We present the interpretation of the following set of parameters, (1) λ_1, λ_2 indicate the user preference to explore. If λ_1 is low then the user likes to explore even after observing their target product attribute values in the results. If λ_2 is high then the user likes to explore for longer during the session.
(2) α_{K_1},α_{K_2}, α_{K_3},α_{K_4} are the weights of different knowledge sources the user might use while reformulating the query. If α_{K_1} is high it implies that the user uses popular/frequent words that are used to retrieve the required attribute values, a lower value indicates that the user has lower *Background knowledge* or does not use this knowledge while querying. If α_{K_2} is high it indicates that the user effectively uses the *Knowledge learnt* from the product space during the search session. The *Knowledge learnt* is especially useful for removing words in query reformulation. If α_{K_4} is high that means the user uses words that do not belong to the product space and instead utilizes similar words.

4.3 LEARNING USER MODELS FROM SEARCH LOGS

CSUM model can learn from search logs by fitting the model to the session data to learn parameters. The search sessions can be used to instantiate IN and K. The search log can be used for evaluating the model. In the following, we describe how the search log is utilized for all the functionalities. The search log we used in this work is obtained from the Walmart E-com search log.

4.3.1 Initial State model

Given the search task information which is target product T and initial query Q_0 and the search log, we initialize the initial IN and K.

Information need initialization

The initial IN is built using the target product T, the starting query Q_0 and the background information learned from search logs. The attributes values of the target product T are used to make binary probabilistic distributions $\theta_{T_{fixed}}$, $\theta_{T_{explore}}$ (θ_{T_f}, θ_{T_e} for short) for *Fixed IN* and *Explore IN* respectively. For attributes in T, $p(V_{A,a_i}|\theta_{T_f} = 1$ and $p(V_{A,a_i}|\theta_{T_e} = c_e$ where c_e is less than 1.

The existing query logs in the same category of T are used to make a background IN θ_b. The common query words indicate the popular preferences of users. The background IN is computed in two ways, θ_{b_f} which only has probabilities belonging to attribute values of T and θ_{b_e} has probabilities belonging to all attribute values. *Fixed IN* (θ_{T_e}), *Exploring IN* (θ_{T_e}) are smoothed using the background models θ_{b_f} and θ_{b_e} respectively. The smoothing parameter (β) is generally very small.

The first query Q_0 provides important information about the initial IN of the user and is used to estimate $P(f_A)$ for all A, the method is equivalent to estimating maximum likelihood solution to generate Q_0. $P(f_A)$ is proportional to the likelihood of attributes of Q_0 from θ_{T_f}, and similarly $P(e_A)$ proportional to the likelihood of attributes of Q_0 from θ_{T_e} and then they are normalized such that $P(f_A) + P(e_A) = 1$.

Knowledge of the product space

The *Background knowledge* (K_2) is initialized in the following way. Using the existing search logs, the translation probabilities $P_{kb}(cat|w_q)$, $P_{kb}(w_p|w_q)$ are estimated. $P_{kb}(cat|w_q)$ is computed based on the co-occurrence of w_q and *cat* in the user sessions which implies that w_q is among the query words used in a session while *cat* is the category attribute of the final product purchased in that session. Similarly. Similarly, $P_{kb}(w_p|w_q)$ is computed based on the number of times w_q is in the final query of a session and w_p is in the words of attribute values of the final product purchased.

4.3.2 Fitting using Search logs

By fitting the model to search log sessions, we evaluate the performance of model description and methods in imitating a real user in a search session, and thus its ability to simulate real users.

We estimate optimal model parameters that explain the search session. We consider $\{\{\alpha_{K_i}\}, \lambda_1, \lambda_2\}$ as the model parameters to be estimated specific to the sessions with the same target product. The best parameters are estimated using query reformulation action and click action. The optimization function tries to maximize the likelihood of the real user actions for both query reformulation and clicks and the model parameters are found accordingly. We estimate some of the parameters globally using all the sessions, $\{c_e, \beta_1, \beta_{kupdate}, \beta_{iupdate}\}$. We assume them to be same for all users and all information needs.

For a query reformulation, we consider the actual words removed or added to the query and learn parameters that optimize the score of those words added to the query and minimize the score of those words removed from the query. For click action, we optimize the average pairwise similarity between actual clicks and clicks generated by the model. We used *jaccard similarity* here.

It is difficult to imitate the complete real user session from beginning to end, an error made in one step will be carried on to the next. Therefore, to simplify we predict the query/click at time t by considering interactions until time t-1 as known and considering the query at time t-1 as the first query for reformulation.

4.4 EXPERIMENTS

We evaluated the model in predicting user actions which are query reformulations for the user's next query and clicks on the result list given the final target product and the initial query as the input.

We explain some implementation details first. The general parameters are estimated using a validation set of search log sessions and they are estimated as, $c_e = 0.5$, $\beta_1 = 0.3$, $\beta_{kupdate} = 0.5$, $\beta_{iupdate} = 0.1$. The threshold for click action is set to 0.05. For simplification, we consider that the user is not exploring beyond the product category, so $P(f_C) = 1$ for all experiments but $P(f_A)$ changes for brand and title attributes. All the functions used in the update model in Section 4.2.3 are formulated in the following way,

$$\begin{aligned} f(\lambda_2) &= (4 + 20 * \lambda_2) \\ P_o(f_A) &= \lambda_1 * \frac{\sum_{o_i \in \{o_i\}} P(V_{A,\{o_i\}} | fixed\_IN)}{\sum_{a_i} P(V_{A,\{a_i\}} | fixed\_IN)} \end{aligned} \tag{4.10}$$

where $\{o_i\}$ is the set of observed attribute values among the results.

$$\begin{aligned} f_1(r_p, clk_p) &= (\frac{1 + 4 * clk_p}{1 + 0.2 * r_p}) \\ f_2(w_p, r_p, clk_p) &= (\frac{c(w_p, p)(1 + 4 * clk_p)}{\sum_{w'_p \in p} c(w'_p, p)(1 + 4 * clk_p)}) \end{aligned} \tag{4.11}$$

where r_p and clk_p indicate rank of the product p and boolean whether the p is clicked respectively, and $c(w_p, p)$ is count of w_p in p.

Finally, to estimate the best parameters for $\{\{\alpha_{K_i}\}, \lambda_1, \lambda_2\}$, we use a grid search approach varying each parameter between $\{0.1, 0.3, 0.5, 0.7\}$.

We estimated the model parameters and evaluated the model on search sessions by its precision and recall in identifying query reformulating words and similarity of the generated clicks to the actual clicks. We evaluated using the new test sessions by estimating parameters on train sessions with the same final target product. From the results, we observed that the model performs reasonably well and also generalizes to a large set of sessions.

4.4.1 Evaluation metrics

Based on the $true\_edits$ (the actual removed and added words to the query), the precision and recall are computed for both removing and adding words, we compute Rem_p@2 (precision for removing words), Rem_r@2 (recall for removing words), Add_p@2 (precision for adding words), Add_r@2 (recall for adding words). To evaluate clicks, for each result page, we calculate Click precision (Clk_p), Click recall (Clk_r), $Clk\_num = (\frac{1}{1+|true\_clicks - generated\_clicks|})$ and average similarity between the actual clicks and generated clicks, Clk_sim. We remove the target product (final purchase product) among the clicks for computing the above metrics as the target product is the given input to the model and the user likely clicks on the target product many times which deviates the click performance.

4.4.2 Experiment Results

From Table 4.1, it can be observed that our defined model $CSUM$ performs better than all other variations. This shows that modeling the update of the user IN is crucial in user modeling and the hypotheses described in Section 4.2.3 are appropriate to model the update. Additionally, we show the results for train and test sessions from four different categories. From the results, it can be inferred that the performance on train and test sessions is almost the same for the query reformulation action. For the click action, the performance is slightly less for the test sessions. We infer that the model can generate a given real user action to a reasonable extent and it is also generalizable in that given a similar new session, it can also approximately generate those session interactions.

4.5 ANALYSIS OF USER BEHAVIOR PARAMETERS

In the following, we verify if the model parameters correlate with the corresponding user behavior characteristics as described in Section 4.2.6.

Table 4.1: Model performance with different variations. Model performance on train and test sessions from 4 categories D1,D2,D3,D4.

Models	Rem_p@2	Rem_r@2	Add_p@2	Add_r@2	Clk_p	Clk_r	Clk_num	Clk_sim
Model_1 on training sessions	0.424	0.5425	0.2447	0.3725	0.5682	0.6319	0.5264	0.7096
Model_1 - IN update on training sessions	0.412	0.518	0.2386	0.3604	0.5682	0.6319	0.5264	0.7096
Model1 on D1 with 296 train sessions	0.3893	0.4954	0.2749	0.3225	0.5525	0.8638	0.5515	0.8037
Model1 on D1 with 131 test sessions	0.3903	0.5067	0.2489	0.3024	0.4825	0.7727	0.4886	0.7192
Model1 on D2 with 504 train session	0.3956	0.4941	0.3139	0.4044	0.5685	0.8892	0.5599	0.7754
Model1 on D2 with 231 test sessions	0.3764	0.4706	0.2931	0.3483	0.5511	0.7885	0.532	0.7344
Model1 on D3 with 173 train sessions	0.4502	0.5537	0.2996	0.3808	0.5133	0.699	0.4461	0.5748
Model1 on D3 with 83 test sessions	0.4013	0.4982	0.2695	0.3794	0.4529	0.5262	0.375	0.5057
Model1 on D4 with 142 train sessions	0.402	0.4806	0.3123	0.4006	0.4755	0.8521	0.4537	0.7545
Model1 on D4 with 62 test sessions	0.3669	0.4351	0.2735	0.3617	0.4847	0.7596	0.467	0.6929

The parameter (λ_1) is inversely proportional to the exploration behavior of the user, to indicate whether the user is exploring other products other than their target product. We compute heuristic measures indicating the exploration behavior of the user in search sessions based on observable actions of the users in the sessions. The user exploring other than their target products (i.e, IN is close to *Explore IN*) is referred to as exploration behaviour and the user focussed to purchase a product instead of exploring is referred to as fixed IN behaviour (i.e, IN is close to *Fixed IN*).

1. User may click or view multiple products which differ compared to the final purchased product.
   $Exploring\_with\_clicks(Ec) = \frac{\sum(1-jaccard_sim(clicked,final))}{no.of clicked products}$ where $jaccard_sim$ is jaccard similarity.

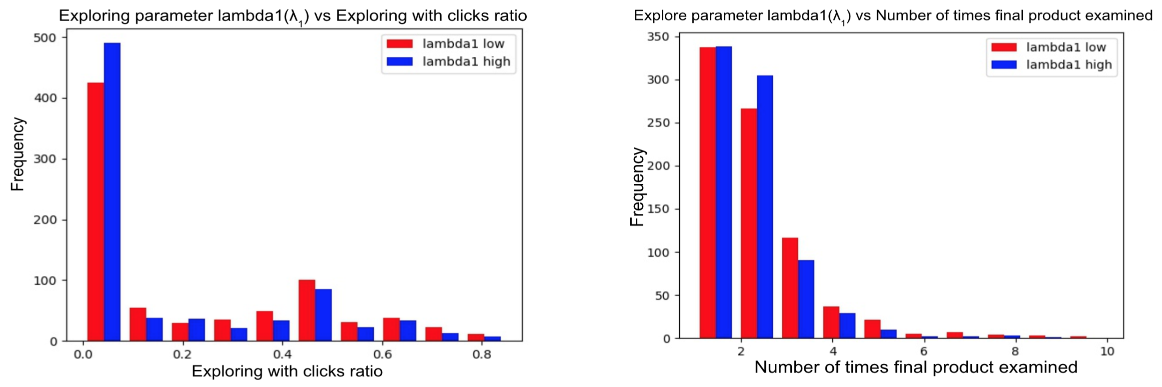


Figure 4.2: Histogram of Ec, EnD for high and low λ_1

2. While exploring, the user may view their final target product multiple times before purchasing it.
   $Exploring\_with\_not\_deciding(EnD)$ is no.of times the final product is viewed before purchasing.

if λ_1 is low we expect the user is exploring and therefore the Ec, End should be higher and vice versa. Figure 4.2a, 4.2b shows the histograms of Ec, EnD for λ_1 being low (red) and high (blue). For smaller values of Ec, EnD, λ_1 high (blue) dominates, which means it is more frequent that when Ec, EnD are low the λ_1 is high. For higher values of Ec, EnD, λ_2 low (red) dominates which implies λ_2 will be low when Ec, EnD are high. Therefore, the histograms confirm our hypothesis that λ_1 should be inversely proportional to Ec, End and show that indeed λ_1 is correlated to the exploration behavior of the user.

4.5.1 Analysis of user behavior patterns using the model

We use the estimated parameters of the cognitive state of the model on the search log to reveal and understand any user behavior patterns. As we have six parameters: $\{\lambda_1, \lambda_2, \alpha_1, \alpha_2, \alpha_3, \alpha_4\}$, each session can be represented as a data point in a six dimension space where this representation is from the user behavior perspective.

Variance in user behavior

In general, different users searching for the same product can have variations in their behavior, and the same user can have different behaviors while searching for different products. Does the behavior vary more across users or products?

The variables var_u, var_p indicate the average variance of all parameters across users and products respectively. We also analysed the variances along each parameter for some users/products and all users/products. The average variance of all parameters is high across the same user sessions compared to the same product sessions. Further, the variance of λ_1, λ_2 is generally higher indicating users differ more in fixed IN vs. exploration behavior. Therefore, we conclude that on average the variance in user behavior purchasing the same product is less than the variance in the behavior of the same user buying different products.

Common patterns through clustering

All the sessions are represented using the six parameters and are clustered. We used K-means clustering. Each cluster centre is interpreted as a behavior pattern representing that cluster. Clustering is done with different number of clusters like from 2, 3 to 32 and the cluster centers are shown in Table 4.2. 1. With just 2 clusters, only λ_1, λ_2 varies indicating most distinguishable user behaviors patterns are exploration vs. fixed IN behaviors. 2. With 3 clusters, α_{K_3}, α_{K_4} also differ in cluster centres, thus the next major difference in user behavior is in different type of knowledge source used for making queries which is using only product space words(K_3) vs. using only similar words outside product space (K_4).

Table 4.2: Cluster centers

ClusterId	Cluster size	λ_1	λ_2	α_{K_1}	α_{K_2}	α_{K_3}	α_{K_4}
0	799	0.659	0.133	0.392	0.38	0.404	0.340
1	779	0.139	0.635	0.413	0.407	0.269	0.400
0	513	0.607	0.159	0.385	0.433	0.151	0.475
1	687	0.111	0.686	0.425	0.408	0.286	0.385
2	378	0.628	0.156	0.386	0.315	0.679	0.202
0	86	0.105	0.669	0.651	0.672	0.128	0.130
8	85	0.102	0.693	0.643	0.159	0.148	0.662
5	76	0.695	0.103	0.179	0.142	0.695	0.147
3	75	0.105	0.695	0.161	0.159	0.129	0.657

4.6 CONCLUSIONS AND FUTURE WORK

We proposed a novel cognitive user model CSUM for E-Com search, which is an interpretable generative model that simulates how a user might behave in a search session when searching for a particular product. It goes beyond the existing work by explicitly modeling and updating the cognitive state of a user, including the user's information need, background knowledge, and new knowledge learned in a search session. Given a particular product to be searched for, the generative model simulates how a user would formulate, re-formulate a query as needed, and click on results, with interpretable parameters that can be estimated using a search log.

The CSUM model is interpretable in that the query reformulation or click actions generated by the model are explainable in terms of which components of the user model lead to the action. We can further analyse how each knowledge component and IN contributed to the action and thus analyse why the action has been taken. In addition to the explainability, all components of the CSUM model are related to different user behaviors, and search strategies and the parameters of the model corresponding to each of the component models correlate with those behaviors. Thus the CSUM model can be controlled by varying the parameters, and different model parameters enable the simulation of different types of users. For example, the exploration parameters can be manually varied to simulate queries and clicks, which mimic users that explore more before making any purchase. The knowledge component parameters can be varied to simulate users that have expert knowledge vs poor knowledge regarding product domain vocabulary. In this way, the model interpretability is useful for generating variation and simulating different types of search sessions.

The CSUM model has many applications. One of the applications that we have explored is to use as a tool to mine search log data to analyze and understand user behaviors buried in the search log. Because of its in-depth modeling of the cognitive states of users, CSUM enables us to reveal interesting user behavior patterns that cannot be extracted using existing user models. Our preliminary evaluation shows several interesting findings regarding user behavior. For example, we show that there is a greater variation in a user's behavior across different products than the variation in the behavior of different users searching for the same product, suggesting that it is likely more effective to learn from past users who searched for the same/similar product to help a current user than to learn from a user's past search history (on different products) to improve the user's current search session. We also reveal that a major factor causing variations in user behavior is to what extent the user wants to explore the product domain, which reflects a user's clarity about the information need, suggesting that it is important to provide customized search support based on the inferred exploration parameter of the model. While these findings are already useful, the model can be used to support many other kinds of analysis (e.g., analysis of various knowledge representation component models and how they evolve in a session), which would be interesting future work.

In addition to it, we further evaluated the model in predicting user actions like query reformulation and clicks given the final target product and the initial query. By initializing the IN and knowledge, the initial query can also be simulated. Thus another major application of the CSUM model is to use it as a user simulator based on the search log data, which is needed for the quantitative evaluation of interactive search engines as well as training reinforcement learning algorithms for optimizing E-Com search. It would be highly interesting to explore such future applications of CSUM.

The CSUM model leverages the user search log to estimate the parameters, but the model cannot learn complex search patterns from the search log. Additionally, we propose query reformulation utilizing the starting query and the information need. Further, the model does not predict whether a query or click action should be taken next. Thus, in our next chapter, we propose a unified supervised user model ILUM (Imitation Learning based User Model) to simulate a complete user model for E-commerce search by generating queries, clicks and purchases along with simultaneously deciding the next action type. The supervised model is based on imitation learning, and the model can learn complex search patterns from a large number of search sessions over a wide range of information needs, thus extending the CSUM model.

Chapter 5: IMITATION LEARNING FOR USER SIMULATION

In this chapter, we explore supervised user modeling based on imitation learning, which attempts to learn from user search sessions from search logs. We aim to address the limitations of many previous works including PRE and CSUM models which cannot leverage the real user data to build an end-to-end user model simulating all the user actions.

5.1 INTRODUCTION

Previous works, including PRE query formulation framework, have proposed and studied user simulation methods, mostly using publically available datasets or using manually curated datasets of user search sessions, which have very limited variations in user information needs (INs). For example, TREC Session track datasets are utilized for training or evaluating user simulation methods in multiple works [14, 39]. However, these datasets only have hundreds of real user sessions over tens or hundreds of information needs (IN); thus, the existing work has only modelled limited user INs. To overcome this limitation, we propose to leverage large-scale user search logs to build user simulation models, which enables us to learn from a wide range of information needs. The CSUM model utilizes search logs however it is limited to only learning the interpretable parameters, the knowledge and IN language models. The CSUM model does not take complete advantage of the user-generated data i.e., it cannot learn more complex search patterns of users.

We propose a supervised user model based on imitation learning, which attempts to learn from the user data in the search logs which we refer to as Imitation Learning based User Model (ILUM). We focus on studying a particular IL method (Behavior cloning) and propose a deep-learning framework for implementing it. We utilize large-scale user search log data, which enables us to model a wide range of information needs and evaluate on a larger test data than the previous works, which better evaluates the generalizability of the model. We utilize the search log data to learn a complete user simulation model ILUM and propose an imitation learning based framework for training the model. Imitation learning has been studied for training simulators for multiple phenomenon [50, 54, 55], but it has not yet been explored for building a user search simulation model. The learning algorithm we propose is a behavior cloning algorithm, an IL method, and we propose an RNN architecture for the ILUM model. We also adapt an existing probabilistic-based user simulation model [39] in order to analyse the learning behavior of the ILUM model. We analyse the advantage and drawbacks of the ILUM model by using the probabilistic model as a baseline.

In this work, we utilized the E-commerce (E-com) search log data as we had access to it. However, the ILUM model we proposed can be easily extended to learn from the web search log data as well. We evaluated both simulation models by simulating three standard search actions in an E-com search process, 1. Query, 2. Clicking on a result 3. Purchase a product. We evaluated the ILUM model by computing the accuracy and similarity of the model in simulating the next action given the actions until that time. Experiments show that the ILUM model performs slightly better than the unsupervised probabilistic method overall with a statistically significant difference. We further analyze the models for predicting different actions. The ILUM model performs better on purchase actions but is worse than the probabilistic model on querying and clicking actions. Although the ILUM model has lower performance compared to the probabilistic model to simulate/predicting the first actions in a session it performs much better for generating consequent actions in that session based on the context of previous actions and search engine results. These results show that **the ILUM model learns to predict actions that are coherent with the previous actions in the session and are based on the given task.** The complementary benefits of the unsupervised model and the ILUM model from our experiments indicate that both of them can be potentially combined for a hybrid model with even better performance.

In this chapter, we make the following contributions. The ILUM model learns from a large set of information needs and more users than any existing simulation models by utilizing search log data. We propose a supervised imitation learning approach for user simulation. Imitation learning is not studied for user search simulation yet; thus, our work makes the first attempt at it. The ILUM model opens up multiple possibilities for further research, including applying advanced IL algorithms like inverse reinforcement learning and extending the current model by exploring different representations of the task, search context, optimization metrics/loss functions, and candidate actions. Search log data also enables learning from real users to build user search simulators including building different simulators for different user behaviors and different categories; thus enabling evaluation of IIR systems using a large number of user simulators learnt from user search logs, our work makes a contribution in this line.

5.2 RELATED WORK

The existing work on user simulation in IR has mostly proposed unsupervised methods for simulating different user actions [39, 41, 42, 43, 65]. An unsupervised method is studied for building a user model to analyse user behavior patterns of E-commerce users using E-comm search log data [34]. Although the previous models are interpretable and based on intuitive heuristics, they cannot take advantage of real user sessions to learn from them. Thus, in contrast to previous methods, in this work, we propose a supervised simulation model that learns from the search log.

The supervised learning methods are less studied partially due to the unavailability of user search session data. A learning-to-rank method is proposed for click simulation by Carterette et al. [14] utilizing the TREC session track dataset. Click models are also learnt from user search log data [14, 60] and are used for simulation. In this work, we make further steps and propose a complete supervised user simulation model for different user actions utilizing a larger dataset of user search sessions to learn patterns from the user search logs. Further, user search simulation is mostly studied as separate models for each user action in the search process, but in this work, we attempt to learn a unified model for user actions. While the decomposed models may be easier/less complex to build from heuristics or learn compared to a unified model, a unified model for all user actions is more general in that all actions can be learnt and improved together, especially a unified model compares different types of actions like query, click, stopping the session for taking the next action, thus simultaneously learns deciding the action type as well.

Search log data is widely utilized to query analysis, user modeling and analysis [72, 76, 77, 79]. These models make query suggestions, model user intent given a session, or try to predict the next action of a user [119] given a partial user session. However, these models are not generative models that can simulate sessions mimicking the real users given a task or an intent as input to the model. There are many click models proposed utilizing the search log [60, 120] to evaluate or train an IR system and relevance feedback algorithms. These methods, however, can only simulate clicks, whereas we study a complete user model to simulate all standard user actions in E-commerce search.

Imitation learning is studied for simulation in multiple different fields like robotics, self-driving vehicles [50], computer games and human-computer interaction [54, 55]; however, it has not been studied before for simulating search users in information retrieval. The closest related work in this context is simulating user dialogue systems using inverse reinforcement learning [56, 57, 58], but these works generate dialogue in conversation and are not applicable for simulating search actions. Imitation learning is useful when it is hard to specify a reward function for performing a task; instead, expert data is available for learning. IL is well suited for search simulation because in the case of search simulation, the user search sessions act as expert data or demonstrations. Further, it is hard to specify the reward function optimized by real users during the search process.

The evaluation of a user simulation model also has many open challenges, one of them is that the ground truth of the user model is unknown. Several evaluation methods are proposed and utilized in previous works, most of them compare the simulator-generated data to the real user-generated data using different metrics like similarity, session statistics and so on [34, 35, 39]. The user simulation models are also evaluated by testing the performance of IR systems with the simulators, similar to indirect evaluation [14, 89]. In this work, we evaluate the ILUM model by comparing the simulator-generated actions with the actual user actions and computing the similarity and accuracy of the model.

5.3 PROBLEM STATEMENT

A search session generally comprises the following interactions, the user makes a query as an action, and the system gives a ranked list of search results. Then the user responds with another action where the possible actions could be either clicking on a search result or reformulating the query, or so on, and this changes the response from the search engine, either by producing a new ranked list or showing the clicked result details.

5.3.1 Task

A user searches because of their information need or preference of products in case of an E-commerce search. Thus we assume that the user actions are guided by their task or information need. Simulation implies simulating a real phenomenon which in this case is a user searching using an IR system to satisfy/fulfil their information need or task. Thus a user simulation model should be able to simulate a search session performing the given task.
Challenges in search logs: In existing user simulation studies in web search, a text description in the form of questions to answer [110] is provided as the task or information need. Thus, the user's task is clearly provided for the simulation model to imitate a user.

While the search log data contains a large number of user search sessions, users' task in these sessions is unknown. Thus in order to use the search log for learning or evaluation of user simulation, the user task should be estimated for a given session. Thus an additional prior step is required to compute a user task for a given session.

Estimating a user task from a session is similar to the problem of computing user intent from a search session [121]. However, the user task should be interpretable in order to be able to generate new search sessions with new tasks or manually creat tasks. One of the major applications of the simulation model is to conduct controlled experiments which need a controlled variation or selection of tasks; thus, the task should be interpretable. Further, a user task also need not be too specific; different sessions can have the same task and given a task, different possible sessions could be generated performing it, whereas the user intent methods [121] identify an intent that is specific for each session.

In this work, as we are studying E-commerce search simulation, the task can be modelled as preferred products or product attributes. For example, given a session, the purchases and clicks can be utilized to find the preferred product attributes and the task can be defined accordingly. Similarly, only purchases can be used for a more specific definition of the task. In this work, we model the task as a specific purchase task, where only the purchases made in the session are used to define the task.

5.3.2 Actions

Users make many different actions while interacting with an IR system which also depends on the interface of the IR system. For example, while searching on an E-commerce website, the user may search for a query and/or choose filters, click on recommendations or similar results, click on a product, add it to the cart or go back, or purchase different variations of the product. However, for learning a user simulation model, we start by simplifying the user browsing behavior, i.e., by assuming a basic IR interface.

We assume a simplistic interface for the IR system where the search system takes a user query as input and returns ranked results as output. The user can either make a new query or click on one of the results. If a user clicks on a result, the system then shows the details of that result on the same page as the ranked results; thus, the user can either purchase that clicked result or click on another product from ranked results as their next action. The user can change their query at any time by making a new query.

We consider three standard actions that a user can make,

1. query action - the user can make a query to the search system
2. click action - the user can click on the ranked list presented by the search system
3. purchase action - the user can purchase a product at time t after clicking on that product at time $t - 1$.

Therefore, with the basic interface we described earlier, the user can make a query action followed by a query or click or purchase action and a click action followed by a query or click or purchase action. The user can make a purchase action only followed by a click action, and the first action is always a query action.

5.4 IMITATION LEARNING BASED USER MODEL (ILUM)

Imitation learning is especially useful when the reward function is hard to specify for performing a task, and instead, it is easier to obtain expert data for performing a task, which is likely the case with user simulation because the reward function optimized by real users is unknown and hard to specify for most tasks such as search process in information retrieval. Thus in order to simulate search users, the search log data can be used as expert data or expert demonstrations.

Imitation learning involves learning from expert demonstrations in order to imitate expert behavior in a given task. The expert demonstrations form the training data and are often in the form of human responses/actions for different situations/states of the environment relevant to the task. For example, in the case of search, the human actions are from the users and the state of the environment is the search engine's response. The states of the environment along with human actions can also form a sequence of state action pairs, referred to as trajectories. For example, user search sessions in IR are a sequence of interactions between the user (human responses) and search engine system (environment), each search session can be treated as a trajectory. Let S denote the set of states and A denote the set of actions. IL aims to learn a policy or an action model for making an action given a state. $p(A) = \pi_\theta(s)$, where π_θ is the policy, $p(A)$ could be a distribution over different actions.

Different IL techniques learn the policy in different ways using expert demonstrations; behavior cloning [122, 123] and inverse reinforcement learning [55, 124] are the two major techniques. In this work, we explore the behavior cloning method for learning. Our aim in this work is to study the effectiveness of learning for user search simulation using search log data; therefore, we leave the question of the best imitation learning method to future work.

In Behavior cloning, the policy is learned by optimizing a loss function on the user/expert data composed of (state, action) pairs. Let a trajectory (sequences of (state, action) pairs) is denoted as $(s1, a1), (s2, a2), ...(sn, an)$ where si is the state in S and ai is the action in A. Behavior cloning assumes an i.i.d of the (state, action) space i.e., the (s, a) pairs of a trajectory are treated as independent of each other. The loss function is thus aggregated over all (s, a) pairs in all trajectories as follows,

$$argmin_\theta \sum_{(s,a)\in G} L(a, \pi_\theta(s)) \tag{5.1}$$

where G is the set of all (s, a) from all trajectories.

5.4.1 Behavior cloning for user search simulation

Imitating search users involves simulating a search process by interacting with a search engine or IR system. User search log data contains real user interactions with an IR system where the sequence of interactions is grouped as a search session. In this case, the search engine system is the environment, and the real user makes the human responses.

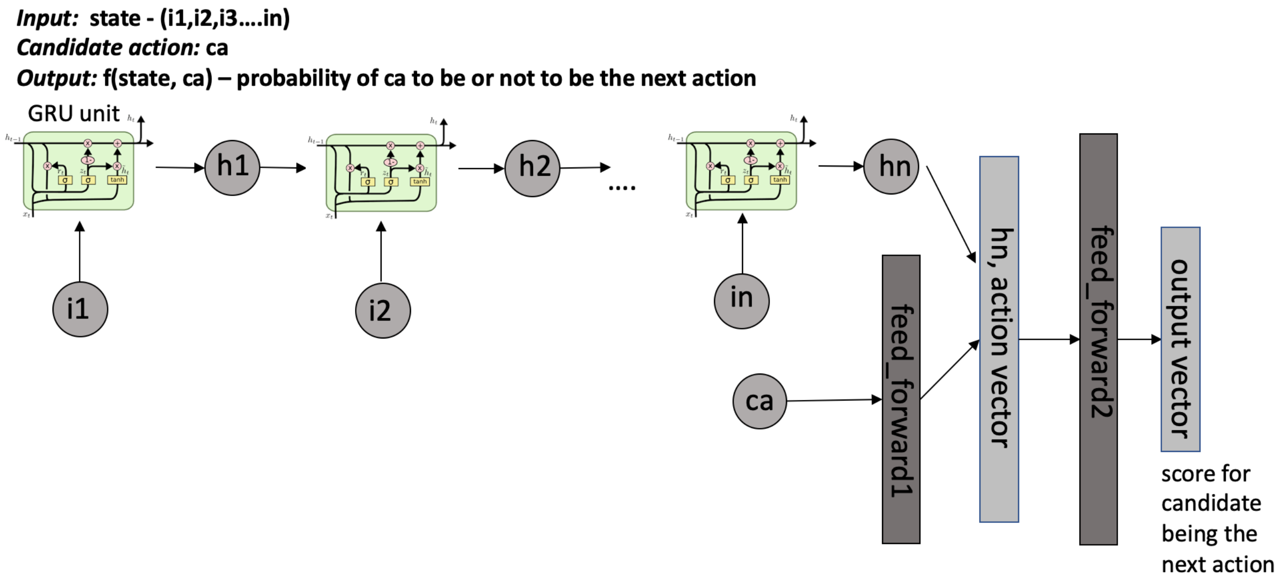
We consider the same basic IR interface as described in Section 5.3.1. The ranked list of results produced by the search system and the clicked page details shown as a response for a click are part of the state of the environment. As the user actions are guided by the user task, we define the state of the environment to also include the user task. Finally, as behavior cloning assumes an i.i.d of the state, action space, we include the previous interactions in the session as part of the memory of the state of the environment. Additionally, as we are utilizing E-commerce search log data in this work, we propose the user simulation model for the product search simulation; however, it can be similarly utilized for other information search scenarios as well.

Thus we describe the user search simulation problem in terms of a behavior cloning problem in the following way, Let the user task/product need is represented as a set of product items that the user likes denoted by n. Let A define a set of actions, and Ap denote the set of action types and each action is associated with an action type in Ap. Let a_t denote the action taken by the user at time t and the r_t is the response/output of the search system for action a_t. The state of the environment at time t includes interactions until time $t-1$ and the user product need n. Therefore, the s_t can be defined as a sequence of interactions denoted as $(i1, i2....i_{t-1})$ where $i_j = g(n, a_j, r_j)$ and g is the aggregate function combining different components of the interaction. Finally, each search session in the search log is a sequence of (state, action) pairs $(s_1, a_1), ...(s_m, a_m)$.

For each state s, there is a set of possible actions that a user can make, and a set of candidate actions C are chosen from them. The model learns a function f to predict $f(s, a)$, the probability of an action a for a given state s. Let (s^*, a^*) be the actual state-action pair, the loss function for action $a \in C$ is computed as $L(a^*, f(s^*, a))$, the loss function may also be computed as a ranking loss over all actions in C which is $L(a^*, R(f(s*, a)|a \in C))$ where R is a ranking function.

5.4.2 Learning model

The learning model essentially predicts the action with a given state input, thus the network architecture of the model can be adapted from the action prediction models [119] or user intent model [120]. In this work, we propose a simple architecture for learning the model. We propose an RNN-based framework for the model function f since the state input is a sequence of interactions. The state s is sequence of interactions denoted as $(i1, i2....in)$. Given the candidate action vector and interaction vectors, f predicts the score of a candidate action to be the next action for the given state. Our proposed network is outlined in Fig 5.1.



Input: state - (i1,i2,i3....in)
Candidate action: ca
Output: f(state, ca) – probability of ca to be or not to be the next action
GRU unit
h1
h2
....
hn
i1
i2
in
ca
feed_forward1
hn, action vector
feed_forward2
output vector
score for candidate being the next action

Figure 5.1: Model architecture

where ca is a candidate action and *output vector* gives the probability that ca will be the next action or not. The model uses a GRU network for learning the hidden state from the sequence of interactions, the GRU unit can be replaced by any RNN unit.

5.4.3 State

As mentioned above the state s_t is the sequence of the interactions until time $t-1$ given by $g(n, r_0, a_0), ...g(n, r_{t-1}, a_{t-1}))$. Note that, a_t is the action taken at time t and r_t is the response of the search engine for the action a_t. The first state s_1 is given by $g((n, r_0, a_0))$ where r_0 and a_0 are empty or null as there are no prior interactions and n is the user task, and a_1 is a query made by the user as the search process always starts with the user making a query.

The response of the search engine includes representation of both the list of ranked lists shown at time t and also the clicked product/result details if the user clicked on a result, i.e., if the action a_t is a click. Finally, if a_t is a query, the response r_t is the list of ranked results. If a_t is a click, the response r_t is both the list of ranked results from the previous response and the clicked product details. If a_t is a purchase, then r_t is the list of ranked results from the previous response.

5.4.4 Loss function

The optimization function compares the actual action made by the user at time t to the candidate action. We can define it as a *cost* function of a candidate action *ca* and actual action $a*$.

$$L(f(s^*, ca)) = cost(s^*, ca, a^*) \tag{5.2}$$

where (s^*, a^*) are the actual state-action pair, the *cost* is the cost function. The *cost* function can be defined in multiple ways, like the similarity between candidate action and actual action or their correlation according to the search log, or whether they are same or not and so on. In this work, we start with utilizing a binary function for *cost* which is 1 or 0 depending on whether $ca == a^*$ or $ca \neq a^*$ respectively.

5.4.5 Candidates

We train the ILUM model to predict among query actions, click and purchase actions on products for a given state.

As mentioned in the problem statement, at each time t based on the state s_t, there are possible actions that a user may make. These actions are useful for training the model on both positive and negative samples where some actions are correct/good, or some actions are incorrect/bad with respect to the user's actual action and the state st.

As the session always starts with a query first, the possible candidates for a_1 are queries. The possible actions at any time t include:

1. click action on each product of the ranked results of r_{t-1}
2. candidate query reformulations.
3. purchase action of a product p if a_{t-1} is a click on p.

As we cannot train a model for all possible actions, we sample candidate actions from them. For example, we sample top-k candidate queries relevant to the task n as candidate queries for the initial query. Similarly, candidate click actions and query reformulations are also sampled.

Negative sampling: Among the candidate queries, it is likely that candidate query reformulations belonging to similar tasks are similar to each other. The ranked list also likely contains similar products in it. Therefore the candidate actions may be very similar to each other and the actual action; thus, it may be hard to differentiate for predicting the actual action. Therefore, we add more candidates which are very different from the target action, and these actions act as purely negative samples.

Thus, we sampled candidate queries that are dissimilar to the task. For example, if a session corresponds to a particular product type, we sample among popular queries of other product types as negative candidate queries as the user is very unlikely to make those queries. Similarly, we choose products belonging to product types other than the session's product type as negative samples for clicking.

5.5 EXPERIMENT SETUP

In the following, we discuss the dataset details and implementation details of the simulation model and the metrics we have used for evaluating the user simulation models.

5.5.1 Unsupervised Probabilistic model

We propose an unsupervised probabilistic model as a reference or baseline model to compare with the ILUM model. We analyse the advantage of learning for the ILUM model in predicting user actions and its drawbacks as well through our experiments. The probabilistic method is unsupervised, i.e., it does not learn from the search log. We propose different models for simulating the query, click and purchase actions and a separate decision model to decide which action to make next.

For the query model, we adapt the simulation methods proposed in Maxwell et.al [39]. This method can be understood as one of the instantiations of the PRE as it involves instantiations of the model of precision and the model of recall combined in the form of a topic language model and collection model similar to our instantiations in Chapter 3. A query is generated based on the language model of the task or product need following the QS3+ algorithm [39]. We compute $p(w|\theta_t)$ as the language model of the user's preferred products. In this work, we define $p(w|\theta_t)$, based on the task, as the language model of the purchased products in the given session. A query is generated in the following way,

$$argmax_Q(\frac{1}{nQ}\sum_{w} p(w|Q)\frac{p(w|\theta_t)}{p(w|\theta_C)}) \tag{5.3}$$

where nQ is the length of the query Q, $p(w|\theta_C)$ is the collection language model, $p(w|\theta_t)$ is language model of the preferred products and $p(w|Q)$ is language model of a candidate query.

For the click model, at time t the ranked list of results produced at $t-1$ are examined in order of their position. A product is clicked, i.e., click action is taken based on the likelihood of click for that product given by $p(w|p\_clk)$. The $p(w|\theta\_clk)$ denotes the probability of the product based on previously clicked products, $p(w|\theta_t)$ and $p(w|\theta_C)$ are the task and collection language models. If the probability is greater than a threshold μ_c, the click action on the product is taken; otherwise, not.

$$p(w|p\_clk) \propto \lambda * ((1-\alpha) * p(w|\theta\_clk) + (\alpha) * p(w|\theta_t)) + (1-\lambda) * p(w|\theta_C) \tag{5.4}$$

We extend the click model to propose a purchase model which is relevant for E-commerce search. The likelihood of the product being purchased uses previous purchases (computed as $p(w|\theta\_pur)$), $p(w|\theta_t)$ and $p(w|\theta_C)$. If the previous action is a click on the product p, then the clicked product is examined for purchase using $p(w|p\_pur)$. Similar to the click action, we use threshold μ_p for making a purchase action.

$$p(w|p\_pur) \propto \lambda * ((1-\alpha) * p(w|\theta\_pur) + (\alpha) * p(w|\theta_t)) + (1-\lambda) * p(w|\theta_C) \tag{5.5}$$

For deciding the next action type, if the previous action is a click, the product is examined for purchase; if it is not purchased according to $p(w|p\_pur)$, then the ranked list is examined from the position after the previous clicked product, and any relevant product is clicked based on $p(w|p\_clk)$. If the ranked list is not interesting to continue to explore, then the query action is taken making a new query. Multiple strategies are proposed to decide whether to continue to explore. We utilize the fixed depth strategy [125] in which y products in the ranked list are examined for clicks irrespective of their relevance; then a new query is posed. Finally, if the previous action is a query, then the ranked list is examined from the start for a click. If the previous action is a purchase, then the ranked list is examined from the position after the purchased product is in the list for further clicks.

Note that, the click and purchase models of this probabilistic model involve updating the purchase likelihood and click likelihood based on previous purchases and clicks respectively. This algorithm is similar to updating the knowledge state as it is essentially updating the information need for every click or purchase. In the PRE framework, we study the knowledge update of relevant and irrelevant documents for query reformulation, whereas in this work the knowledge state update is used for the click and purchase models.

Ideally the probabilistic model is learning from the search history in the session to make better predictions for later (consequent) actions in the session. In the experiments, we analyse whether the ILUM model can effectively leverage the search context through training and perform even better for predicting later actions.

5.5.2 Dataset

As mentioned in Section 5.1, we utilized E-commerce search log data for our experiments. As the search sessions can involve searching about various different products and user search behavior may highly vary across product types, we considered a few product types to start with. We use a query classifier to assign a product type to the queries of a session. As users may also search for different product types in one session, we divide a session to contain queries belonging to only one product type that are all searched on the same day. Thus each of the resulting sessions belongs to a certain product type. The product types considered are the following: *coffee maker*, *television*, *speakers*, *shirt*, *shoes*,*chair*,*electronic cable*,*coffee*,*vitamin*, *pet food*, *shampoo*, *notebook computer*.

We sample sessions with at least one purchase as we are modeling users with purchase tasks. E-commerce search log data consists of many different user actions, so we filter other actions or filter sessions with those actions. For example, we merge add-to-cart actions with purchases together. We ignore sessions with clicks on recommendations, similar results or with purchases on variations of a product after clicking it. Therefore we filter the search session to only contain the query, click and purchase actions.

There can be more than one purchase in a session, thus a session need not end with a purchase. All the purchases in the session are aggregated to make the task representation n and the language model $p(w|\theta_t)$.

We sampled 10716 sessions from the search log. For training the ILUM model, this dataset is split into train and test sessions, we made a stratified split so that the distribution of sessions of different product types remains the same in train and test sessions. The statistics of the train and test data are shown in Table 5.1.

Table 5.1: Dataset statistics after sampling for sessions from 14 product types, where sessions included a purchase, and after filtering actions other than queries, purchases and clicks.

	No.of sessions	No.of (state,action) pairs	No.of Query actions
Train	8959	62423	19097
Test	1757	11949	3595

We represent the query and product using pre-trained word embeddings. The pre-trained word embeddings are learnt using separate models for query words and product words. Thus, query embedding is computed as the average of pre-trained embeddings of query words. The product embedding is computed as the average of pre-trained embeddings of words in the product title.

5.5.3 State and Action representation

We utilize the query and product embeddings to represent a state and an action in the ILUM model input.

Action representation: An action can be either a query or clicking on a product, or purchasing a product. If an action is a query action, we compute the query embedding, If an action is either a click or purchase action, we compute the product embedding on which the click or purchase action takes place. Therefore, the action a is represented as follows,
query_embedding(a)$\| \overrightarrow{0} \| \overrightarrow{0}$ if a is a query action,
$\overrightarrow{0}$ $\|$product_embedding(a)$\| \overrightarrow{0}$ if a is a click action,
$\overrightarrow{0} \| \overrightarrow{0}$—product_embedding(a) if a is a purchase action.

Thus the action vector is of the dimension: query emebdding dimension + 2*product embedding dimension, the $\overrightarrow{0}$ is a zero-vector of the corresponding embedding dimension.

State representation: A state s constitutes responses of the search system, actions taken and the task.

The user task embedding is based on products purchased in that session. Thus, we compute task n vector as aggregated embedding of the products purchased in that session. Note that the task needs to be identified from the search session to guide the user simulation model. So, n is considered constant for the entire session.

The response r_t of the search engine has two components, a ranked results list and a detail page of a clicked product depending on whether there is a click in the previous action. The user may look at all results on the response page for deciding the next action, thus we aggregate the embeddings of the products in the ranked list to represent the ranked list. Second, we use product embedding to represent the detail page of the clicked product. Thus, the response embedding r_t of the search engine is represented as,
$Avg\_product\_embedding$(ranked list)$\|$$product_embedding(a_t)$ if a_t is a click,
$Avg\_product\_embedding$(ranked list)$\| \overrightarrow{0}$ if a_t is not a click,
where $product\_embedding(a_t)$ is the embedding of the product clicked as part of action a_t

We model the function g of the state as a concatenating function. Finally with the action embedding a_t and response embedding r_t, the interaction at time t is represented as follows,

$$n\ embedding \| a_t\ embedding \| r_t\ embedding$$

5.5.4 Candidate actions

As described in Section 5.4.5, the candidate actions for a given state include candidate queries, candidate products for click action, and a candidate product for purchase action.

For choosing candidate queries, we use the product type of the session as the user likely makes queries of that product type. Thus, we count the frequency of a query in the sessions of a product type and choose the top 1000 frequent queries of that product type as total candidate queries. We also include all the queries of the same product type in our sampled dataset in the candidate queries set. To decrease the time complexity of the model, we choose a few candidates randomly as the final candidate queries set.

For a given state, the possible clicks are all the products in the ranked list of the previous response, i.e., for state s_t, the products in the ranked list of r_{t-1} are the candidate click actions. We considered the top 30 products of the ranked list and randomly sampled some of them as candidate clicks to reduce the time complexity of the model. For a given state, there is sometimes a candidate purchase action that is possible, if a user's previous action is a click on a product i.e., if a_{t-1} is a click, then that clicked product say p_{t-1} will be the candidate for purchase action.

Note that for the first state in a session, the candidate actions only have candidate queries. Additionally, while training and testing the model we always include the actual user action among the candidate actions.

5.5.5 Negative sampling

In addition to the candidates described in Section 5.5.4, we also sample additional actions which are highly unlikely to be the user's next action. Therefore, among the candidate queries set, we also include a few randomly sampled popular queries of product types different from the product type of the session. Similarly, for clicks, we sample products from other product types and add them to the candidate clicks.

5.5.6 Random simulation model

In our experiments, we used a *Random model* as a reference to compare the performances of the ILUM model and the Prob model (short for probabilistic model). The random model chooses five random actions from the set of candidate actions as the top 5 predicted actions. It also randomly chooses one action among the top 5 as the top 1 predicted action.

5.5.7 Evaluation metrics

We evaluate the model for predicting the next action given a state. The simulation models produce a ranked list of candidate actions for predicting the next action for a state. First, we compute the accuracy of the model in predicting the action. We compute *Accuracy@k* by taking top-k actions in the ranked list as follows,

$$Accuracy@k = \sum_{(s,a)\in G} \frac{(a \in top\_k\_pa)}{|G|}, \tag{5.6}$$

where G is the set of all (s,a) pairs that are in the dataset to be tested, top_k_pa is the set of top-k predicted actions by the model according to its ranking.

Simulating the user search process implies generating actions which look like they are produced by a human user, thus the model need not predict exact actions correctly (as the exact user actions may vary a lot). *Accuracy@k* only computes exact matches between actions, however, the aim of the simulation model is to mimic users and make actions that are realistic; therefore, the simulation model can also predict similar actions to user actions. Thus, we also compute the similarity between the predicted action and the actual user action as an evaluation metric. We consider the top-ranked candidate action as the predicted action by the model.

$$Avg\_action\_sim = \frac{\sum_G similarity(a,pa)}{|G|}, \tag{5.7}$$

where pa is the predicted action by a model. The similarity function used here is the cosine similarity between the respective embeddings, so if a, and pa are both query actions, it computes the similarity between query embedding. If a, and pa are both click actions, then it computes the similarity between product embeddings and similarly for purchase actions. If a, and pa are not of the same action types (for example, if a is a query but pa is a click), then the similarity is 0, $similarity(a,pa) = 0$.

For analyzing the model further, the above metrics are also computed for each action type separately. For example, we consider only (s,a) pairs where a is a query action and compute the accuracy of the model in predicting query actions.

$$\begin{aligned} Accuracy\_A@k &= \sum_{(s,a)\in G_A} \frac{(a \in top\_k\_pa)}{|G|}, \\ Avg\_action\_sim\_A &= \sum_{(s,a)\in G_A} \frac{(a \in top\_k\_pa)}{|G|} \end{aligned} \tag{5.8}$$

where A can be query or click or purchase type and G_A is the set of all (s,a) pairs where type of action a is A, $action\_type(a) = A$.

Finally, we also computed the model accuracy in predicting only the next action type correctly, i.e., the actual action and predicted action should be of the same action type but need not be the same action. We compute this metric for all action types
$Action\_type\_acc = \frac{\sum_{(s,a)\in G}(action\_type(a)==action\_type(pa))}{|G|}$

where A can be query or click or purchase action type, pa is the top predicted action in the ranked list and $action\_type(a)$ gives the action type of a .

5.5.8 Parameters

We utilized pre-trained word embeddings for representing queries and products. We choose the following hyperparameters of the ILUM model for our experiments. The model architecture is shown in Fig 5.1 whole details are given below,

1. GRU : hidden size = 50, number of layers = 1
2. $feed\_forward1$: $Linear() \rightarrow RELU() \rightarrow Linear() \rightarrow dropout()$
3. $feed\_forward2$: $Linear() \rightarrow RELU() \rightarrow Linear()$

For choosing candidate actions for a state, we randomly sample 10 queries from all the candidate queries as described in Section 5.5.4. Similarly, we randomly sample 10 products from the top-30 ranked list of products for candidate actions. The number of candidate purchase actions is only 1 or 0 depending on the previous action. For negative sampling, we additionally sample 10 random queries and 10 random products for clicks from other product types during training only.

The unsupervised *Prob model* is initialized with the following hyperparameters, $\alpha = 0.7, \lambda = 0.9$, and $y = 10$, λ is a smoothing parameter and is therefore chosen as a higher value. The thresholds are $\mu_c = 0.05$, $\mu_p = 0.3$. The purchase threshold is higher so that only products very similar to the task are chosen, whereas the click threshold is lower because the user may still click on similar products.

5.6 EXPERIMENT RESULTS

Through the experiments, we aim to analyze and compare our proposed imitation learning model (ILUM model) and the unsupervised probabilistic model (Prob model). We obtain the performance for both models by computing all the evaluation metrics described in Section 5.5. We want to answer the following research questions through the results,

1. Which of the models performs better for different evaluation metrics?
2. How does the performance of the models vary for different types of actions?
3. Does the ILUM model has any advantage from learning? What is the learning behavior of the ILUM model on training data?

In our experiments, we evaluate the ILUM model with two variations, the ILUM model either with negative sampling or without negative sampling for candidates. As explained in Section 5.4.5, we expect that negative sampling should improve the model performance. Additionally, we also evaluate the Random simulator method as a reference for comparison.

First, we evaluate the model's accuracy in predicting the user action for a given state. We compute *Accuracy*@*k* for each instance in the test data with 11949 state-action pairs and compute the average. The ILUM model predicts the score for each candidate action for a given state, and a ranked list of actions is generated according to the score. We compute *Accuracy*@1 and *Accuracy*@5 as described in Section 5.5.7 by taking top-1 and top-5 ranked actions to evaluate if the user's actual action is among them. Similarly, the accuracy of the Prob model and random simulator is also computed on the test data. The results are shown in Table 5.2. As *Accuracy*@*k* only computes exact matches, we also evaluate average action similarity, Avg_action_sim, which computes the similarity between the actions. The average action similarity between the predicted action (top-1 ranked action) and the actual user action for a given state is shown in the third column of Table 5.2.

From Table 5.2, it can be inferred that the ILUM model performs better at Accuracy@5 whereas the Prob model performs better at Accuracy@1. The ILUM model also performs better than the Prob model in terms of Avg_action_sim. We perform an independent t-test for computing statistical significance between the performances using the p-value as 0.01. Table 5.2 indicates that the ILUM model performs better than the Prob model with statistical significance and the Prob model performs better at Accuracy@1 with statistical significance as well. The random baseline results act as a reference, and all the models perform better than random with a statistical significance, which indicates that they perform reasonably well. Overall from the results, we infer that both the ILUM model and the Prob model

Table 5.2: Accuracy@1, @5 and action similarity of the models in predicting next action, * - statistical significance between ILUM model (w neg sampling) and Prob model

Model	Accuracy@1	Accuracy@5	*Avg_action_sim*
ILUM model	0.3265	0.4299	0.7306
ILUM model (w neg sampling)	0.3452	0.5354*	0.7737*
Prob model	0.4010*	0.4844	0.6275
Random baseline	0.0109	0.2202	0.5659

perform reasonably well and are comparable. Further, adding negative samples during training has greatly improved the ILUM model's performance, i.e., the ILUM model with negative sampling performs better than without. The ILUM model with negative sampling ("w neg sampling" for short) performs slightly better than other models.

Table 5.3: Evaluation of accuracy of the models for each type of action separately

Model	Accuracy_Q@5	Accuracy_C@5	Accuracy_P@5
ILUM model	0.5032	0.0748	1.0
ILUM model (w neg sampling)	0.3883	0.3829	1.0
Prob model	0.5591	0.4709	0.4175
Random baseline	0.3241	0.1725	0.1810

Table 5.4: Evaluation of action similarity of the models for each type of action separately

Model	Action sim_Query	Action sim_click	Action sim_purchase
ILUM model	0.9683	0.9463	1.0
ILUM model (w neg sampling)	0.9656	0.9551	1.0
Prob model	0.9702	0.9826	1.0
Random baseline	0.9459	0.9494	0

We further analyze the performance of the models for predicting each type of action separately to analyze their performances deeply. For example, we compute *Accuracy_Q@k* with all (state, action) pairs where the actual action is a query. Similarly, the average action similarity is also computed for each action type as *Avg_action_sim_A* where *A* is the action type. From the results in Table 5.3 and Table 5.4, we observe that the ILUM model with negative sampling has improved performance for clicks but decreased performance for queries compared to the model without negative sampling, even though the same number of negative samples are included for both candidate queries and clicks. Further study is needed to understand this result.

Table 5.5: Accuracy of the models in predicting only action type correctly.

Model	Action type Acc_overall	Action type Acc_Query	Action type Acc_Click	Action type Acc_Purchase
ILUM model	0.7532	0.7185	0.6434	1.0
ILUM model (w neg sampling)	0.7966	0.6239	0.8014	1.0
Prob model	0.6395	0.7324	0.6975	0.4175
Random baseline	0.5966	0.5051	0.9781	0.0

Table 5.6: Accuracy and Action Similarity results with Training and test data using ILUM model(with and without negative sampling)

Model	Accuracy@5	Action sim_overall
ILUM model on training data	0.4287	0.7306
ILUM model on test data	0.4299	0.7410
ILUM model(w neg sampling) on training data	0.4601	0.7737
ILUM model(w neg sampling)on test data	0.5354	0.7377

Additionally, the ILUM model performs best in predicting purchase actions, whereas the Prob model predicts query and click actions better. As described in Section 5.3.1, we consider the purchases made in a session as the user task in that session and give the task as input to both models. The task representation n is an aggregated representation of user purchases in a session in the ILUM model, and $p(w|\theta_t)$ is the language model of purchased products in the Prob model. The ILUM model learns to predict purchase actions better than the Prob model because of learning from the purchase action instances in the training data, as it highly matches the task n which is part of the state input. The benefit is that the task can be varied for systematic variations in simulated sessions using the ILUM model; however, there is also a drawback in that the model is strictly guided by the task, especially for purchase actions. Thus different and better variations of task representation, for example, a broader representation of task in terms of product attributes, are pivotal for building an effective ILUM model; this is also one of the challenges for further research in this direction.

One of the reasons that the model may not be performing well in predicting queries is that we computed queries and product embeddings in separate embedding spaces. The state representation is mostly composed of aggregated product embeddings in the form of the task, search results and clicked products; therefore the model may be predicting higher scores for product actions resulting in the query actions being ranked lower.

Finally, we also evaluate the action type accuracy metric as described in Section 5.5.7. The results are shown in Table 5.5 for predicting only the action type correctly, i.e., predicting a query type action when the actual action type is a query or a click when the user's action is a click type. The evaluation for predicting only the action type indicates the model performance as the decision model deciding the next action type.

The ILUM model performs much better than the Prob model overall for predicting the action type. The action type accuracy is higher for purchase, confirming that the ILUM model is better for predicting purchase actions. The action type accuracy of the ILUM model for clicks is also higher than the Prob model, which could be again because the state representation is mostly composed of product embeddings, thus predicting click actions more often. The Prob model has thresholds for determining whether to click or purchase. The lower performance of the Prob model for click type and purchase type predictions could be because the thresholds are probably too high resulting in fewer click and purchase type predictions. We have not tuned the thresholds for the initial study of our models. Fine-tuning the hyperparameters may improve the performances of both models and should be part of future work.

Next, we analyze the learning behavior of the ILUM model on the training data. The train and test performances are shown in Table 5. We observe that the training and the test data performance are close to each other. The training performance is not very high, which is possible because of the heavy class imbalance in the dataset. The binary loss function considers only the actual action as the correct action. Therefore for a given state, the model should predict all the candidates except one as negative actions (low probability for being the next action. Thus among the (state, action) instances, there is approximately a 1:40 ratio for positive to negative actions, as there are about 41 candidate actions for each state (20 queries, 20 clicks, 1 purchase) according to our parameters listed in Section 5.5.8. Thus the high-class imbalance results in the model mostly learning to identify the wrong actions correctly, which itself leads to a very low loss value. Therefore, the accuracy for predicting the correct action (or identifying an actual action as the next action) may still be lower for the training data. Thus, we conclude that improving the optimization function is important for further developing the ILUM model for user simulation. For example, considering the similarity between actions or computing ranking metrics for the ranked list of candidate actions similar to learning to rank as loss functions are some of the possible alternatives for further exploration.

To further analyze the learning behavior of the ILUM model, we analyse the accuracies and similarities at each timestamp in a session i.e, for each state from s_1 to s_n in a session. For each timestamp, we compute the average $Accuracy@k$ and Avg_action_sim, which enables

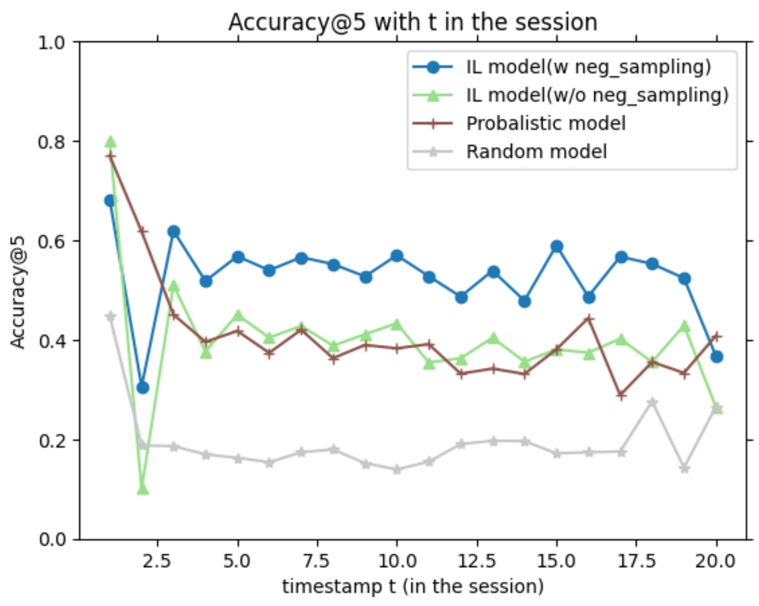


Figure 5.2: Accuracy@5 with t in session

us to study how the performance of the model varies at different points/times in a session for predicting the next action. The dataset consists of many sessions of shorter length and fewer and fewer sessions of longer length because longer sessions are generally less frequent. In the test data, we computed performance at each timestamp in a session, and the performances of different models are shown in Fig 5.2 and Fig 5.3 for the metrics *Accuracy*@5 and *Avg_action_sim*, respectively. As there are very few sessions longer than length 20 (less than 50), the average performance is computed on very few samples for those timestamps; thus, we analyze the performance only for timestamps until 20. From the figures, it can be inferred that the ILUM models perform very well for the later timestamps in the session and outperform the probabilistic and random models. The ILUM model (w neg sampling) has the highest performance among them. The ILUM performance is lower from timestamps 1 and 2 because initially, there is not enough context in the state except the task, and thus it results in lower performance. At the later timestamps, the state contains more context of the search session, and along with the task, the models learn to predict better. Thus we can infer that the ILUM model indeed learns from the search interactions in the session for making the next action. There is an initial drop in performance at timestamp 2; one of the potential reasons could be that the previous action at timestamp 1 is a query (the first query), and because there is not enough context, it may influence the next action to be likely predicted as query whereas most sessions generally have the second action as click followed by the initial query. Similar trends are also observed for *Accuracy*@1. Finally, it can be inferred that the supervised ILUM model has the advantage of learning while predicting the consequent

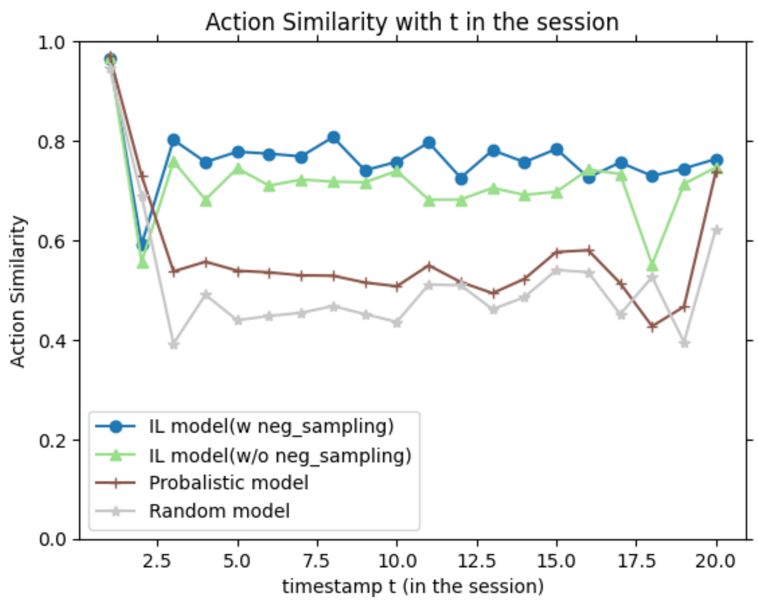


Figure 5.3: Average action similarity with t in session

actions in a session because the state representation improves over time with more previous interactions and the model learns better from previous actions. The probabilistic model, on the other hand, has decreased performance for predicting later actions in a session. But it is better at simulating the beginning or first actions (at timestamps 1 and 2) using the given task. The complementary benefits of the models indicate that a hybrid approach combining the ILUM model with an unsupervised method is also an interesting research direction and may lead to even better simulation models.

Finally, we also analyze the performance of the models for different product types (*pt*) or categories, i.e., we separate sessions of different product types (based on queries in the session) and compute the average performance for each *pt* accordingly. From Table 5.7, we observe that most category performances are in the same range, except the categories *SHOES* and *VITAMIN*, which have the highest performance of the ILUM model, implying that there may be more search patterns in those categories for the ILUM model to learn. We analyzed the test data, especially the query actions where the ILUM model does not predict the correct action. We observe that the ILUM model generally predicts broader queries over specific queries; it performs better for broader queries and does not perform well for specific queries. Some of the examples are shown in Table 5.8. For example, the query "french press" and "desk chair" are broader queries and are predicted correctly. Specific queries like "5 1/4 car speakers", "dunkinâ€™ donuts colombian coffee cake up", and "fancy feast natural treats" are not predicted correctly; instead, for these queries, the ILUM model predicts very broad queries in their respective categories. The reason for predicting broad queries is maybe

Table 5.7: Performance (Accuracy@5) of the models for different product types

product types	ILUM model	ILUM model(w neg sampling)	Probabilistic model
Coffeemaker	0.4035	0.4058	0.5099
Electronic cable	0.4413	0.5810	0.4431
Pet food	0.4139	0.5957	0.5028
Coffee	0.4066	0.4082	0.5148
Speakers	0.429	0.3793	0.4688
Television	0.481	0.4557	0.5569
Vitamin	0.4195	0.6372	0.4755
Shampoo	0.4672	0.4555	0.5644
Chair	0.4451	0.4204	0.4884
Notebook computer	0.4118	0.4608	0.4509
Shirt	0.4809	0.542	0.4733
Shoes	0.6667	0.6458	0.3542

because they match the task and the previous search interactions better overall than specific queries.

Table 5.8: Queries predicted by the model

Actual queries	Predicted top-k queries
'french press'	'french press', 'ninja coffee station', 'keurig k 400', 'keurg coffee maker, 'teal 8 cup coffee maker'
'5 1/4 car speakers'	'bluetooth outdoor speaker','radio bluetooth','sound bar with remote control', 'soundbar', 'soundbucket'
'dunkinâ€™ donuts colombian coffee cake up'	'coffee pods' , 'folgers coffee k cups', 'peets decaf k cups'
'fancy feast natural treats'	'treats for dogs', 'i love dog food', 'dog chicken jerky', 'canned dog food 12 pack', 'dog bones for large dog'
'desk chair'	'desk chair', 'lounge chair for kids', 'toddler saucer chair', 'backpacking chairs with extendable legs', 'camp chair for beach'

In conclusion, the ILUM model indeed learns to perform better than the Prob model, especially for purchase actions and at the later timestamps of a session. The Prob model performs better for query actions and the beginning or initial actions of a session due to a better query model. Both models have complementary benefits and can also be combined for future models. This is a preliminary study of the proposed ILUM model, and it is already shown to perform reasonably well. The model can be further improved for predicting query actions potentially by utilizing a combined embedding space for queries and products. The optimization metrics utilized in the model can also be varied. For example, the loss function can be the similarity between actions instead of binary measure. The task representation controls the variation of the simulated actions of the ILUM model, and better representation may allow for better variation in actions making the simulations more realistic.

5.7 CONCLUSION AND DISCUSSION

Our conclusions in this work are fourfold: 1) we make the first attempt exploring imitation learning for training a complete user search simulation model by leveraging search log data which is shown to perform reasonably well; 2) Both the ILUM model and the probabilistic model are comparable and have advantages and disadvantages compared to each other, they can be potentially combined for a better model as part of future work; 3) The ILUM model learns to simulate actions better using the search log data; it learns the search context from previous interactions to make the next action and performs better as the search context increases i.e., for consequent actions in a session, the model also learns to simulate purchase actions better given the task. But predicting the query actions better, especially at the beginning of a session can further improve the ILUM model. 4) Finally, more advanced IL models should be used for further development in building user simulation models. We describe these directions in detail in the following.

First, we propose a new user simulation model in IR using imitation learning by leveraging large-scale E-commerce user search log data, and this is the first work studying this problem. We propose an RNN-based framework for training a behavior cloning model, a particular IL method, using the search log data. We utilized the search log data which enables learning and evaluation using a large number of information needs than the previous literature. The existing work has mostly used manually curated datasets [39] or TREC datasets [14, 19], which only have hundreds of information needs, whereas in this work we utilized 10000 sessions with different purchase tasks from different product types, thus modeling and evaluating many INs.

We also proposed a probabilistic method for simulating E-commerce search users, adapted from the previous work. The ILUM model and probabilistic model are evaluated on their accuracy, and action similarity and the experiments show that both models have comparable performances, with the ILUM model with negative sampling performing slightly better. Although we compare the models, both models are just different methods to utilize the search log, and they are not necessarily competing. Indeed, both models may be combined for even better performance. For example, the ILUM model can be combined with the probabilistic approach by including likelihoods as input features, or the ILUM model can mainly decide the next action type leaving the action generation to the probabilistic method as the ILUM model has shown better performance for predicting just action type correctly. This is another research direction with the potential for great improvement.

Third, from our analysis, we inferred that the ILUM model performs best for predicting purchase actions associated with the task, which means that the ILUM model can simulate a session with a given task which is required for simulating sessions with different tasks for offline evaluation. The ILUM model performs better for predicting consequent (at later timestamps) actions in a session, which implies that it learns from the previous interactions to simulate the next action better for a given task. However, the ILUM model performs poorly in predicting queries, clicks and predicting actions at the initial timestamps of a session. This model opens up multiple research directions for further developing IL-based user simulation models; for example, exploring different task representations, and different optimization metrics for building different types of IL-based user simulation models.

Additionally, the evaluation of simulation itself has many open challenges [36, 89], our evaluation metrics are accuracy and similarity, other metrics like the max similarity between the simulated action and any of the relevant user actions [35] and the ranking performance of the search algorithm [14] can also be used. We consider that different action types have zero similarity, but search log data can be used to find the correlation between different actions, which can also be used as a metric. Indirect evaluation through applications [89] like offline evaluation is also another method. Although the user simulation models are not studied for simulating an entire session in this work, the model can be utilized for simulating a session given a set of products as product need for the simulator. Thus further evaluation of the model for simulating a session would indicate the utility of the simulation model for the evaluation and training of IIR systems.

Finally, note that we make a preliminary exploration of the IL approach in this work and more advanced IL models, like inverse reinforcement learning [124], can be used to build a potentially more effective simulation model. The inverse RL model is better for simulating the sequence of actions in a session as it can learn from its mistakes while training, whereas the behavior cloning model only sees the real user's sequence of actions for training.

Chapter 6: A RELIABILITY-AWARE TESTER-BASED EVALUATION FRAMEWORK FOR USER SIMULATORS

In the previous chapters, we have described our contributions to modeling users for user simulation. In this chapter, we address the challenges in the evaluation of user simulation. User simulators can be used to run reproducible IIR experiments which can be controlled and varied systematically to evaluate for different types of IN, different types of users and experiment constraints. However, for any simulator to be really useful, we must be able to evaluate its reliability since if the simulators are not reliable, the conclusions we draw using them to evaluate IIR would be questionable.

Evaluation of user simulation itself has multiple challenges such as variance in user behaviour, unavailability of ground truth, and dependency on the intended use of a user simulator. So far user simulators have been mostly evaluated by assessing how similar a simulator is to a real user using multiple strategies, including comparing simulated and real user session data [39, 44], comparing retrieval performance of IR systems with the real user queries and simulated queries[40, 42, 43, 126], comparison of search session data[14, 45], log-likelihood and perplexity [45]. However, almost all of these evaluation methods require real user search interaction data which is expensive to acquire. Further, those evaluation methods do not directly assess whether the simulator is reliable for comparing IIR systems.

How can we tell whether a user simulator is reliable for comparing IIR systems? This is a challenging question that has not been well studied in the existing work. We attempt to establish a general framework for evaluating the reliability of user simulators for comparing IIR systems. To this end, we propose a novel Tester-based evaluation [36] (TBE) framework for estimating the reliability of a user simulator in a single context of comparing IIR systems. We extend the TBE framework to propose a Reliability Aware Tester based evaluation framework (RATE) to compute the overall reliability of a user simulator and also address the limitations of TBE. The RATE framework is a generalized framework and does not make any assumptions about the user simulation model.

6.1 RELIABILITY AWARE TESTER BASED EVALUATION FRAMEWORK (RATE)

6.1.1 Tester-based evaluation (TBE)

The TBE framework is proposed to evaluate the reliability of a user simulator from the perspective of evaluating IIR systems i.e., it is evaluating predictive validity (as described in [127]) of a simulator in differentiating different IIR systems. The framework introduces a component called Tester to evaluate Simulators.

Tester T: A Tester is a set of IR systems along with an expected performance pattern of the IR systems which is generally an expected order of the performance of the IR systems (e.g., the order of the NDCG scores produced by the two IR systems). To evaluate a user simulator using a Tester, the user simulator first interacts with each of the IR systems in the Tester and the retrieval performance (like the NDCG score) of each IR system is measured with the simulator. The pattern of the performance of all the systems in a Tester can then be compared with the expected performance pattern. The reliability score of the simulator is computed using a measure quantifying how close the two performance patterns are.

For example, if A and B are two IR systems, where an IR system is defined to include three components: retrieval method, document collection, and topics (Information needs). If we have knowledge that system A performs better than system B, a Tester t can be built as follows: $(A, B, A > B)$ which denotes that A, B are two systems and $A > B$ is the expected performance pattern which means A is expected to perform better than B.

Reliability score $F(S,T)$: The reliability score of a simulator S as measured using a Tester T (denoted by $F(S,T)$) is computed as an evaluation measure quantifying how close the obtained performance pattern of S with T and the expected performance pattern of T are. The score $F(S,T)$ is computed in two ways, Success rate(Sr) and Failure rate(Fr). It is considered as *success* or *failure* if the order of the IR systems given by the simulator is correct or incorrect according to the expected performance pattern of the Tester respectively. Continuing our example, if a simulator s interacts with the IR systems in Tester t, and the resultant performance of the system A and system B with s are computed as p_A, p_B for all the topics. If $p_A > p_B$ then it is considered a success (because the expected pattern is also $A > B$); otherwise, it is a failure. Finally, a Success rate (Sr) and a Failure rate (Fr) are computed by aggregating all successes and failures across multiple test cases (multiple information needs).

6.1.2 Limitations of TBE

A major limitation of TBE is that the Testers themselves can be unreliable because the expected performance pattern of a Tester may not be always correct [36]. Indeed, the expected performance pattern is hypothesized using domain knowledge or empirical results of the IR systems, thus the expected performance pattern may not be true when generalized to different information needs, different types of users, or different document collections. The lack of a way to distinguish reliable from unreliable Testers means that TBE cannot yet be applied to accurately measure the reliability of user simulators. It also makes it hard to generate a reliable single reliability score for a simulator when tested with multiple Testers. Thus we extend the TBE framework by considering the reliability of Testers and propose Reliability-Aware Tester-based Evaluation (RATE).

6.2 RATE FRAMEWORK

The main idea of our proposed new framework, named Reliability-Aware Tester-based Evaluation (RATE), is that we evaluate the reliability of the user simulators using Testers which are themselves associated with reliability. We define and formulate both the reliability of the Tester and the reliability of the Simulator such that they can be jointly learned in an unsupervised manner, making it possible to apply the framework to any set of Testers and Simulators without requiring any human supervision.

6.2.1 Reliability of a Tester

How can we compute the reliability of a Tester? Logically, the reliability of a Tester is determined by the certainty of the expected performance pattern. If we have high confidence in the expected pattern of a Tester, we would also have high confidence in saying that a user simulator is not reliable if it fails such a Tester (i.e., fails to show the expected performance pattern), which suggests that the Tester is reliable; similarly, if our confidence in the expected performance pattern of a Tester is not high, we would be less sure whether a user simulator is unreliable even if it fails such a Tester.

Now, how can we assess the certainty of the expected performance pattern of a Tester? One possibility is to empirically answer this question by running the Tester on real users to see whether the same pattern is always observed. However, an IR system may have different performance when interacting with different types of users or search behaviours. Thus the certainty of the performance pattern, also the reliability of a Tester, inevitably depends on the specific users used to estimate it. Thus the definition of the reliability of a Tester is with respect to a group of users. We define the reliability of a Tester as follows.

Reliability of a Tester $R(T)$: *The Reliability of a Tester T is the reliability of the expected performance pattern of that Tester, i.e, the probability that the expected performance pattern is satisfied when tested with a population of users U. Average reliability $R(T)$ is obtained by aggregating reliability scores from different populations of users.*

Formally, if $F(U, T)$ is the probability that the expected performance pattern of T is consistent with the observed performance pattern when the Tester T is applied to a real user U in a population group G. The reliability of T (denoted by $R(T)$) can be defined as the average $F(U,T)$ over all the users in the group.

$$R(T) = \frac{1}{|G|} \sum_{U \in G} F(U,T) \tag{6.1}$$

From Eq. 6.1, one of the solutions for computing $F(U,T)$ by testing the Tester using real user experiments. But obtaining $F(U,T)$ using real user experiments for each Tester T is very expensive and generally infeasible. Thus a solution we propose is to approximate a group of real users with a group of user simulators G'. How well a simulator S approximates a real user U is denoted by $p(U|S)$, which indicates the quality of a simulator. Naturally, when S is a real user, this probability would be 1.

With this approximation, we have

$$\begin{aligned} R(T) &= \frac{1}{|G|} \sum_{U \in G} \frac{1}{|G'|} \sum_{S \in G'} F(S,T) p(U|S) && (6.2) \\ &= \frac{1}{|G'|} \sum_{S \in G'} F(S,T) \frac{1}{|G|} \sum_{U \in G} p(U|S) && (6.3) \end{aligned}$$

Without additional knowledge of specific users that we are interested in simulating, we can reasonably assume that $p(U|S)$ can simply be approximated by $p(\tilde{U}|S)$, where $\tilde{U}$ denotes an "average" user. With this assumption, $\sum_{U \in G} p(U|S) = |G| p(\tilde{U}|S)$, thus we have

$$R(T) = \frac{1}{G'} \sum_{S \in G'} F(S,T) p(\tilde{U}|S) \tag{6.4}$$

As the Reliability of the simulator is also evaluating the quality of the simulator, we propose that $P(\tilde{U}|S)$ is proportional to the reliability of the simulator S, i.e., $R(S)$. Therefore Reliability of the Tester is dependent on the reliability of the simulators in the following way,

$$P(\tilde{U}|S) \propto R(S), R(T) \propto \sum_{S \in G'} F(S,T) R(S) \tag{6.5}$$

6.2.2 Reliability of User Simulator

We now discuss how we can define the reliability of a user simulator $R(S)$. One solution is to compare the behavior of the user simulator with that of a real user and use the similarity to estimate $R(S)$, but we are interested in studying whether it is still possible to define it in some meaningful way even without access to real user data.

One way to achieve the goal is to assume that a simulator is reliable if it gives the expected performance pattern correctly with a reliable Tester, i.e., it has a high success rate, $F(S,T)$

when tested with Tester T. The simulator can be assumed to be even more reliable if it can pass multiple reliable Testers with high success rates. Logically, this is a quite reasonable assumption as it just means that a reliable simulator passes many reliable Testers with high success rates. This heuristic can be potentially implemented in multiple ways. Below we describe two of them.

Symmetric formulation: In this formulation, we compute $R(S)$ as a weighted mean of success rate $F(S,T)$ over a set of Testers, where the weights are the reliability of the Testers.

$$R(S) = \frac{\sum_T F(S,T) * R(T)}{\sum_T R(T)} \tag{6.6}$$

We refer to this formulation as symmetric formulation because note that the reliability of the Tester and reliability of the simulator are both defined in terms of each other as per Eq. 6.4 and Eq. 6.6 respectively. Both formulations are similar and symmetric.

The symmetric formulation gives the simulator a high-reliability score if it passes many highly reliable Testers (high $R(T)$) with high success rates (high $F(S,T)$). Note that $F(S,T)$ and $R(T)$ both range from 0 and 1, therefore $R(S)$ is also between 0 and 1.

Difference formulation An alternative method to compute $R(S)$ is by using the definition of reliability of Tester. The reliability of a Tester is the extent to which the Tester's expected pattern is satisfied by the real user. We can thus assume that a simulator S is more reliable if it best approximates an average user ($\tilde{U}$) in its behavior on Testers. That is, the score of $F(S,T)$ should be close to $F(\tilde{U},T)$ for all Testers. With this notion of reliability, we define $R(S)$ as being proportional to the difference between $F(S,T)$ and $F(\tilde{U},T)$ where $F(\tilde{U},T)$ is $R(T)$ from Eq. 6.1 when all users in G are approximated to $\tilde{U}$. We refer to this as Difference formulation. Formally,

$$R(S) \propto 1/\sum_T (F(S,T) - R(T)) \tag{6.7}$$

$$R(S) = \frac{2}{\sum_T (F(S,T) - R(T)) + 1} - 1 \tag{6.8}$$

$R(S)$ in Eq. 6.8 is scaled such that the value is between *zero* and *one*. Unlike Symmetric formulation, Difference formulation implies that a simulator should have a higher score with highly reliable Testers along with a lower score with less reliable Testers because the difference $F(S,T) - R(T)$ in such cases will be smaller and $R(S)$ will be higher.

Circular dependence of Reliability of Tester and Simulator: In the RATE framework, there is a circular dependency between the reliability of a simulator and the reliability of a Tester which is somewhat similar to the HITS algorithm [128]. In the Symmetric formulation of $R(S)$ (Eq. 6.6), the principle is that a simulator has higher reliability when it satisfies reliable Testers and a Tester has high reliability when it is satisfied by reliable simulators according to Eq. 6.5. The principle in the Difference formulation of $R(S)$ (Eq. 6.8) is that a simulator has higher reliability if it satisfies reliable Testers and does not satisfy unreliable Testers and a Tester has high reliability if it is satisfied by reliable simulators. The circular dependency between $R(S)$ and $R(T)$ captures their relations intuitively and directly suggests constraints that can be interpreted as defining a fixed point solution for $R(S)$ and $R(T)$ jointly. Thus it enables us to learn both reliabilities from all the test values $F(S,T)$ in an iterative way until they converge. Algorithm 6.1 and Algorithm 6.2 are the two iterative algorithms for computing $R(S)$ and $R(T)$ based on Symmetric formulation and Difference formulation respectively. In both algorithms, we start by considering that all Testers are reliable or $R(T) = 1$ which is the assumption of the baseline TBE evaluation approach.

Algorithm 6.1: Iterative algorithm for Symmetric Formulation:

```
Step 1: for each $S$, for each $T$ , Obtain $F(S,T)$
Step 2: Initialize $R(T)^1 = 1$, $n = 1$
Step 3: $R(S)^n = \frac{\sum_T F(S,T) * R(T)^n}{\sum_T R(T)^n}$
Step 4: $R(T)^{n+1} = \frac{\sum_S F(S,T) * R(S)^n}{\sum_S R(S)^n}$
Step 5: $n = n + 1$
Step 6: Repeat Step 2 until the convergence condition is met.
```

Algorithm 6.2: Iterative algorithm for Difference formulation

```
Step 1: for each $S$, for each $T$ , Obtain $F(S,T)$
Step 2: Initialize $R(T)^1 = 1$, $n = 1$
Step 3: $R(S)^n = \frac{2}{(\sum_T |F(S,T) - R(T)^n| * \frac{1}{|T|}) + 1} - 1$
Step 4: $R(T)^{n+1} = \frac{\sum_S F(S,T) * R(S)^n}{\sum_S R(S)^n}$
Step 5: $n = n + 1$
Step 6: Repeat Step 2 until the convergence condition is met.
```

6.3 EXPERIMENTS

In our Experiments, we evaluate the feasibility and effectiveness of the RATE framework. We study three basic Testers by applying them to a set of four representative user simulators whose behavior is known to us. We evaluate whether RATE can reasonably identify reliable simulators and Testers. Specifically, the following questions are answered through the experiments: **RQ1)** Can RATE results distinguish unreliable Testers from reliable ones?

RQ2) Which of the two formulations, Symmetric or Difference formulation is more effective and robust? **RQ3)** How sensitive are reliability scores learnt from the iterative algorithm?

6.3.1 Testers

Query History (QH) Tester: The Tester compares two IR systems where one of them is baseline BM25 and the other is an IIR system that uses previous queries in a session to do query expansion using the method proposed in [129] and improves on the baseline. The expected pattern is that the IIR system should perform better than the baseline. The Tester is denoted by (M, M+QH[α]) where α is the parameter of the query expansion controlling the weight on the current query compared to previous queries. ***Click History (CH) Tester***: Similar to QH Tester, this Tester compares the baseline with an IIR system that uses previously clicked documents to do query expansion and improves ranking over the baseline using the method proposed in [130, 131]. The Tester is denoted by (M, M+CH[β]) where β controls the weight of the current query compared to the clicked history terms. The expected pattern is similar to QH Tester, $M + CH[\beta] > M$. ***BM25 Ablation Testers***: This Tester contains standard BM25 (M) compared with BM25 without TF weighting ($M \setminus TF$) or IDF weighting ($M \setminus IDF$). As the TF and IDF weighting are important components in BM25, the expected pattern for these Testers is $M > M \setminus TF$, $M > M \setminus IDF$ respectively. We used Whoosh [132] library to implement all the Testers.

We implement six Testers for evaluation. We use two QH Testers $\alpha = 0.5$, $\alpha = 0.01$ denoted by (M, M+QH[0.5]), (M, M+QH[0.01]), and two CH testers with $\beta = 0.5$, $\beta = 0.8$ denoted by (M, M+CH[0.8]), (M, M+CH[0.5]). And the final two testers are (M, M\ TF), (M, M\ IDF).

6.3.2 List of Simulators

We leverage the simulators implemented in SimIIR toolkit [63]. The Simulators simulate 4 basic user actions in an IIR session with variable methods supported for simulating each action:

1. **Query simulation methods**: a) *Random query generation (RQG)*, a one-word query is generated which is a random word from vocabulary; b) *Single term query generation (STQG)* a one-word query is generated based on a topic language model; c) *Smarter query generation (SQG)*, two and three-word queries are generated from topic and background language models.

2. **Clicking Snippet decision**: a) *Random click (RC)* generates a random click on one of the results; b) *TREC Qrel click (TRECQc)*, clicks a document only if it is TREC-relevant; c) *Stochastic TREC Qrel click (STRECQc)*, clicks a TREC-relevant document with probability 0.75, and TREC-irrelevant document with 0.35 probability.

3. **Stop browsing the SERP decision**: a) *Random decision (RD)*, randomly decides to stop browsing; b) *Sequential non-rel decision (SnrD)*, stop browsing SERP if 3 consecutive irrelevant documents are seen.

4. **Session Stopping decision** is a *Fixed cost budget (FC)* method where each action requires some cost indicating time or effort spent on that action, and a simulated user is given a fixed amount of cost, thus the session is stopped once the cost budget is reached or when the list of simulated queries is completed.

From these options, we have selected four representative combinations of strategies to obtain four meaningful representative simulated users (see Table6.2) which differ in one or a few components. All Simulators use *Fixed cost budget* for simulating the session-stopping decision.

Table 6.1: List of Simulated users

Simulated User	**Configuration of the User**
Smart Ideal Trec user (SIT)	SQG, TRECQc, SnrD
Smart stochastic Trec User (SST)	SQG, STRECQc, SnrD
Single term stochastic Trec (STST) user	STQG, STRECQc, SnrD
Random User (RU)	RQG, RC, RD

6.3.3 Dataset

The Information needs and document collection are selected from the TREC AQUAINT dataset. A subset of 50000 documents are used as the document collection and 47 randomly selected TREC topics are used as Information needs. The simulators use the TREC topic description and topic judgements as input to simulate the user actions to generate a search session.

We measure the retrieval performance of an IR system with a simulator by computing session-based DCG measure *(sDCG/q)* [133] of the resultant simulated search session. The order of (sDCG/q) scores of two IR systems in the Tester will give whether the simulator has success or not which leads to computing success rate ($F(S,T)$) across multiple topics(Information needs).

Table 6.2: Success rate of the 4 Simulators with six Testers

F(S,T)	SIT	SST	STST	RU
(M,M+QH[0.5])	0.766	0.745	0.809	0.021
(M,M+QH[0.01])	0.723	0.723	0.872	0.021
(M,M+CH[0.8])	0.787	0.723	0.638	0.021
(M,M+CH[0.5])	0.809	0.638	0.575	0
(M,M\TF)	0.34	0.383	0.362	0
(M,M\IDF)	0.532	0.426	0	0

6.3.4 Experiment Results

We applied all six Testers to the four representative simulated users. We evaluate the four simulators, by computing success rate $F(S,T)$ using the six Testers, Table 6.2 shows the results of the success rate scores. The following inferences are made from Table 6.2,

1. Testers $M + QH$ and $M + CH$ can differentiate different simulators and correctly identify the Random User with the least success rate.

2. Query History Tester and Click History Tester shows reasonable behavior.

3. BM25 Ablation Testers did not behave as expected, further experimentation and analysis would be needed to better understand this Tester.

4. Overall it indicates that the SIT simulator as the most reliable and RU is the least which are both reasonable conclusions. This conclusion is reasonable as SIT simulates smarter query generation and ideal clicks and is thus more reliable in comparing IR systems.

By applying the RATE framework, we can now compute the reliability scores of these Testers. Using Table 6.2 results, we apply the iterative algorithm to compute $R(S)$ and $R(T)$. The convergence condition for the algorithm is when the change in $R(S)$ and $R(T)$ is less than 0.00001 and we initialized all $R(T)$ to 1 in *Iteration 0*.

6.3.5 Reliability scores of Simulator and Testers

Tables 6.3 and 6.4 show the resultant R(T) and R(S) scores with Symmetric and Difference formulation respectively.

First, we see that both Symmetric and Difference formulations were able to correctly identify (M, M\TF) and (M, M\IDF) as relatively unreliable Testers consistent with observations in Table 6.2. Although these two ablation testers are reasonable, depending on the set

Table 6.3: R(S) and R(T) using RATE using Symmetric Formulation of R(S)

Reliability of Simulators		**Reliability of Testers**	
Simulators	R(S)	Testers	R(T)
SIT	0.70551	(M,M+QH[0.5])	0.76777
SST	0.65107	(M,M+QH[0.01])	0.76519
STST	0.62699	(M,M+CH[0.8])	0.71432
RU	0.01308	(M,M+CH[0.5])	0.67446
		(M,M\TF)	0.3587
		(M,M\IDF)	0.32689

Table 6.4: R(S) and R(T) using RATE using Difference Formulation of R(S)

Reliability of Simulators		**Reliability of Testers**	
Simulators	R(S)	Testers	R(T)
SIT	0.77963	(M,M+QH[0.5])	0.68797
SST	0.868	(M,M+QH[0.01])	0.68726
STST	0.80399	(M,M+CH[0.8])	0.63722
RU	0.31133	(M,M+CH[0.5])	0.59603
		(M,M\TF)	0.3216
		(M,M\IDF)	0.28395

of information needs and document collection it is possible that the Tester's expected pattern might be incorrect. We also see that the relative order of the Testers according to $R(T)$ scores is the same with both Symmetric and Difference formulations, suggesting that the iterative algorithm is generally robust. Moreover, although we started with R(T) as 1.0 for all the testers, the framework learns different reliability scores for each tester and distinguishes unreliable Testers from reliable ones. We can observe that the Testers with different parameters also have different reliabilities i.e, (M, M+QH[0.5]), (M, M+QH[0.01]) have slightly different R(T) scores and (M, M+CH[0.8]), (M, M+CH[0.5]) have a larger difference in their reliability scores. Thus we conclude that the RATE framework can distinguish reliable Testers from unreliable ones and assign meaningful reliability scores to them.

Second, from the results in Table 6.3 and 6.4, we observe that the reliability scores of Simulators learnt using Symmetric and Difference formulations are very different. Both the formulations correctly identify the random user by giving it the lowest reliability, but the Difference formulation gives a relatively higher score to *RU* which seems undesirable. This is likely because, in Difference formulation, the Random user (RU) is rewarded for lower scores on the two less reliable Testers, (M, M\IDF) and (M, M\TF).

Compared with the Difference formulation, the Symmetric formulation scores appear to be more meaningful, not only identifying the unreliable random user correctly but also identifying SIT as the best simulator, which is quite reasonable as SIT uses smart queries and clicks.

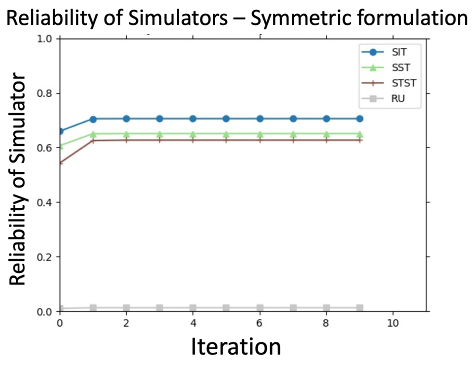


Figure 6.1: R(S) - Symmetric formulation

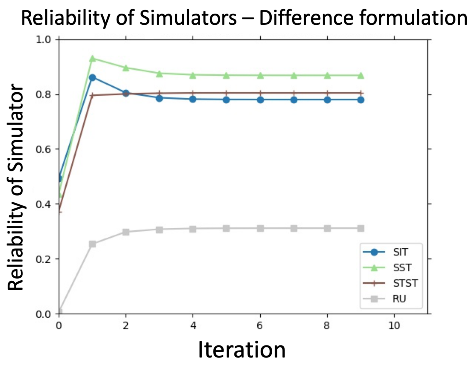


Figure 6.2: R(S) - Difference formulation

Figures 6.1 and 6.2 show the convergence graphs of both algorithms in computing the reliability scores of the four simulators. The values converge fast because there are only a few data points (4 simulators and 6 testers). The convergence graphs further show the difference between both Symmetric and Difference formulations, like the order of the simulators is different in both and RU gains reliability with Difference formulation in Figure 6.2.

To further analyze the robustness of Symmetric and Difference formulations and the RATE framework overall, we modify the experiment by adding a large number of random testers (RUs). An important question we study is that: *Would simulator ranking be affected much by the set of simulators considered?*. We repeat the experiment with 6 Testers and 50 RU simulators along with SIT, SST, and STST simulators. The results are shown in Tables 6.5 and 6.6. The Symmetric formulation gives almost the same R(S) scores as in Table 6.3, but the Difference formulation scores are completely changed where the Random user is now given the highest reliability scores and the other simulators have low reliability. Clearly, the Difference formulation is not robust against changes in the set of simulators especially when more bad user simulators are added. We thus conclude that Symmetric formulation is more effective and robust, answering *RQ2* and partially answering *RQ3*.

Finally, to address *RQ3*, we study the behaviour of the RATE framework by altering the list of Simulators or Testers, including removing one simulator (e.g., 3 Simulators (SIT, STST, RU), 6 Testers) and removing two Testers (e.g., (4 Simulators, 4 Testers(QH testers, Ablation testers)) or (4 Simulators, 4 Testers(QH testers, CH testers)). We only use Symmetric formulation for computing R(S) in all these experiments as it is the preferred formulation. The results are shown in Table 6.7, Table 6.8 and Table 6.9. From the results, we observe that the reliability of a tester is not affected much by changing the set of testers, and similarly,

Table 6.5: Reliability scores with multiple Random Testers - Symmetric formulation

Reliability of Simulators		**Reliability of Testers**	
Simulators	R(S)	Testers	R(T)
SIT	0.7058	(M,M+QH[0.5])	0.5831
SST	0.6515	(M,M+QH[0.01])	0.5812
STST	0.6278	(M,M+CH[0.8])	0.5429
RU	0.0131	(M,M+CH[0.5])	0.5077
RU	0.0131	(M,M\TF)	0.2700
RU	0.0131	(M,M\IDF)	0.2459
...	0.0131		

Table 6.6: Reliability scores with multiple Random Testers - Difference formulation

Reliability of Simulators		**Reliability of Testers**	
Simulators	R(S)	Testers	R(T)
SIT	0.2196	(M,M+QH[0.5])	0.0328
SST	0.2605	(M,M+QH[0.01])	0.0328
STST	0.3119	(M,M+CH[0.8])	0.0317
RU	0.9819	(M,M+CH[0.5])	0.0103
RU	0.9819	(M,M\TF)	0.0057
RU	0.9819	(M,M\IDF)	0.0045
...	0.9819		

the reliability of a simulator is not affected much by changing the set of simulators. For example, the R(T) scores of the QH Testers are the same in Table 6.3, Table 6.7 and Table 6.8; the R(T) scores of CH Testers are also almost the same in Table 6.3 and Table 6.8; and similar results are observed for Ablation testers. Further, the reliability of the simulators SIT, STST and RU is the same as in Table 6.9 and Table 6.3. We also experimented with more such configurations and found similar results. Thus we conclude that RATE is stable and robust in the sense that adding or removing unreliable or random simulators does not affect the evaluation of simulators, and adding or removing testers does not affect the evaluation of the testers. That is, the simulator ranking is not affected much by the set of simulators considered in the RATE framework. This implies that once we have a benchmark set of Testers with their learnt reliabilities, we can evaluate any number of simulators using the RATE framework without much worry about the sensitivity of the simulator reliability scores to the number of simulators participating in the evaluation.

However, we observe that changing the list of simulators affects the reliability of Testers, and changing the list of testers affects the reliability of simulators. This is expected as the computation of $R(T)$ itself is proportional to the average success rate of all the simulators when interacting with T, and vice versa. Interestingly, although the absolute scores are affected, the ranking of Testers is unaffected by the set of simulators which is a positive result indicating the robustness of RATE. Overall, the robustness results indicate that RATE is not highly sensitive to small changes in the list of simulators or Testers.

Table 6.7: Robustness test - Reliability scores after removing CH Testers

Reliability of Simulators		**Reliability of Testers**	
Simulators	R(S)	Testers	R(T)
SIT	0.6423	(M,M+QH[0.5])	0.7678
SST	0.6329	(M,M+QH[0.01])	0.7671
STST	0.6423	(M,M\TF)	0.3588
RU	0.0145	(M,M\IDF)	0.3171

6.4 CONCLUSION

In this work, we proposed a new evaluation framework called Reliability-Aware Tester based Evaluation framework(RATE) to evaluate User Simulators for their reliability in comparing IIR systems. We first propose a novel **Tester-based evaluation approach** for evaluating the reliability of user simulators, in which we would construct a Tester based on a set of IR systems with an expected performance pattern and apply such a Tester to a user simulator to see if the user simulator would generate the expected performance pattern. Testers themselves can be unreliable. Therefore, we extend the TBE approach to study how to quantify the reliability of Testers by proposing a new **Reliability-Aware Tester based Evaluation (RATE) framework**. In the RATE framework, we defined and formulated the reliability of the Tester and consequently the reliability of the simulator where both have circular dependence between them. We propose two methods to compute $R(S)$, Symmetric and Difference formulation and propose an iterative algorithm to learn both reliability scores. Our experiments show that RATE is effective in distinguishing unreliable Testers from reliable Testers and also can distinguish reliable and unreliable simulators effectively. Between the two formulations, the Symmetric formulation is found to be more effective and stable to

Table 6.8: Robustness test - Reliability scores after removing BM25 Ablation Testers

Reliability of Simulators		**Reliability of Testers**	
Simulators	R(S)	Testers	R(T)
SIT	0.7697	(M,M+QH[0.5])	0.768
SST	0.7092	(M,M+QH[0.01])	0.7668
STST	0.729	(M,M+CH_0.8)	0.7122
RU	0.0162	(M,M+CH_0.5)	0.6719

Table 6.9: Robustness test - Reliability by removing SST simulator

Reliability of Simulators		**Reliability of Testers**	
Simulators	R(S)	Testers	R(T)
SIT	0.7098	(M,M+QH[0.5])	0.7789
STST	0.6382	(M,M+QH[0.01])	0.786
RU	0.0133	(M,M+CH[0.8])	0.7097
		(M,M+CH[0.5])	0.6914
		(M,M\TF)	0.3469
		(M,M\IDF)	0.2774

compute $R(S)$. We further observe that RATE is robust in that the reliability score of a simulator is robust against the list of simulators considered and similarly for testers.

RATE provides a foundation for potentially establishing a new paradigm for evaluating IIR systems using user simulation. As an immediate future work, we envision leveraging RATE to establish a novel open evaluation platform where the research community can regularly add more Testers and simulators (which will naturally happen as researchers develop novel Testers and simulators), and the reliabilities of both the testers and the simulators can then be computed using the Symmetric formulation algorithm. Such a RATE platform will, for the first time, enable the use of potentially many user simulators to evaluate IIR systems with reproducible experiments.

Although we have only used unsupervised learning to learn reliability scores, the framework also allows semi-supervised learning, the exploration of which is an interesting future direction. For example, if some of the Testers have the success rate scores when tested with real users ($F(U, T)$), these scores can be used as initial $R(T)$ scores and they can be used to learn the reliability of other Testers and simulators through propagating reliability using the iterative algorithms proposed in the framework. Similarly, if real user data is available to assess the reliability of some simulators, such reliability scores can also be easily incorporated into the framework.

Evaluation of PRE and CSUM models: The RATE framework is generalized and does not make any assumptions about the user simulation model. Therefore, one of the future works is to evaluate PRE and CSUM frameworks using RATE. We can evaluate the user simulators corresponding to PRE and CSUM models just like SIT and SST simulators in the experiment setup in Section 6.3.2. For the PRE query simulations model, the different instantiations of the PRE model can be compared with each other, while having the remaining components of the user simulation constant. With this evaluation, it can be verified whether the PRE framework can be used for query simulation for evaluating IIR systems and which of the instantiations will be effective. The CSUM model can also be evaluated if the E-commerce dataset and set of information needs are available for the simulator to interact with the IR system.

Chapter 7: CONCLUSION AND FUTURE WORK

In this thesis, we addressed some of the challenges in user simulation modeling, leveraged search logs to build user simulation models and proposed methods for evaluating user simulation models. The contributions of the thesis are described in detail in the following,

1. We propose a user model called **CSUM (Cognitive State User Model)** for the E-commerce search scenario, which is an interpretable generative model that simulates user actions in a search session when searching for a particular product. It goes beyond the existing work by explicitly modeling and updating the cognitive state of a user, including the user's information need, background knowledge, and new knowledge learned in a search session. Given a particular product to be searched for, the generative model models how a user would formulate, and re-formulate a query as needed, and click on results, with interpretable parameters that can be estimated using a search log. We used the CSUM model to mine search log data to analyze and understand user behavior patterns buried in the search log. Our preliminary evaluation shows several interesting findings regarding user behavior. We show that there is a greater variance in user behavior across different products than the variation in the behavior of different users searching for the same product, suggesting that it is likely more effective to learn from past users who searched for the same/similar product to help a current user than to learn from a user's past search history (on different products). We also identify that a major factor causing variations in user behavior is to what extent the user wants to explore the product space.

2. Query simulation is a critical component of the user simulation. We propose a novel unified **Precision-Recall-Effort (PRE) optimization framework for simulating query formulation and reformulation** which is applicable for modeling both Web search and E-commerce search users. PRE generates a query conditioned on the current knowledge state of the simulator and updates the knowledge state for simulating the next query during query reformulation, therefore the query formulation and reformulation are modelled uniformly in our work. By instantiating each of the components of PRE (Model of Recall, Model of Precision, Effort, Knowledge Update, and Optimization algorithm), we can not only cover existing methods for query simulation but also derive many interesting new interpretable query simulation methods. Experiment results show that it is necessary to model both recall and precision, thus providing empirical evidence for the rationale of the design of the PRE framework. Further, We show that PRE adaptively restricts query length by generating longer or shorter queries depending

on the information need. PRE can be used as a roadmap for systematically exploring more effective models for simulating query formulation.

3. We make a first attempt at exploring imitation learning for building a supervised unified user simulation model by leveraging user search logs. We model and evaluate using a large number of information needs than the existing work in user simulation which is due to the limited availability of search log data. **We proposed a novel Imitation Learning based User Model (ILUM) by studying a particular ILUM method (behavior cloning) and propose a deep-learning framework for implementing the model.** We evaluate the ILUM model by using it to simulate three standard search actions for E-commerce which are querying, clicking and purchasing. Experiments show that the ILUM model performs slightly better than an unsupervised method overall, with a statistically significant difference. We further analyze the models for predicting different actions. The ILUM model performs better on purchase actions but is worse than the probabilistic model on querying and clicking actions. Further, although the ILUM model has a lower performance to simulate the first actions in a session it performs much better for generating consequent actions in that session based on the context of previous actions and search engine results. This shows that **the ILUM model learns to predict actions that are coherent with the previous actions and are based on the given task.** The probabilistic model has an advantage at the beginning of the session, especially in predicting initial query actions. The complementary benefits of the unsupervised model and the ILUM model indicate that both of them can be potentially combined for a hybrid model with even better performance.

4. We address some of the challenges in the evaluation of user simulation by proposing a novel **Tester-based evaluation framework to evaluate the reliability of a user simulation for comparing IIR systems**. The advantage is that the framework does not necessarily require real user data to evaluate the simulator and it aims to evaluate predictive validity. We further **extend the TBE framework by proposing a Reliability Aware Tester-based evaluation (RATE) framework** to address the drawbacks of the TBE framework. Specifically, the TBE approach did not consider the reliability of the Testers themselves. In the RATE framework, we defined and formulated the reliability of the Tester and consequently the reliability of the simulator where both have circular dependence between them. Our experiments show that RATE is effective in distinguishing unreliable Testers from reliable Testers and also can distinguish reliable and unreliable simulators effectively. We further observe that RATE is robust in that the reliability score of a simulator is robust against the list of simulators considered

and similarly for testers. RATE provides a foundation for potentially establishing a new paradigm for evaluating IIR systems using user simulation. As future work, we envision leveraging RATE to establish a novel open evaluation platform where the research community can regularly add more Testers and simulators.

Advantages and Disadvantages of proposed user simulation models:

All the user models proposed have complementary advantages compared to each other for modeling user actions. The PRE query formulation framework is an optimization framework that is a query formulation algorithm given user knowledge state which is updated over time. The PRE model is ideal for text-based web searches. However, it does not model IN updates and different types of IN characterizing different search behaviors. On the other hand, in the CSUM model, we propose different heuristics for how to update the IN over time and how different types of knowledge may be used during search; additionally, the CSUM model represents IN/user preferences and user knowledge separately making it more suitable for E-commerce search. Thus CSUM model is better for simulating E-commerce searches. We can integrate the PRE and CSUM models by using the PRE optimization algorithm for query formulation and CSUM model for preference model and background knowledge together to make an improved user search model for E-commerce search.

Both the PRE and CSUM models have meaningful parameters that can be varied in order to simulate different user behaviors. However, these models cannot learn search patterns from a user search session. The Imitation Learning based User Model (ILUM) model on the other hand is a data-driven approach which is trained to learn to simulate a complete user model, including different user actions and decisions during the search. The CSUM model also utilizes the search log, but it is limited to only learning the model hyperparameters from the search log and cannot learn more search patterns whereas the ILUM model can learn complex search patterns. The ILUM model learns to simulate an average user in search as it learns from all sessions together, whereas PRE or CSUM models can simulate a specific type of user. We can improve the ILUM model by integrating the PRE precision and recall likelihoods as part of the query representation of the ILUM model and the IN model updates from either PRE or CSUM models can also be included while predicting the next actions. Thus, the interpretable unsupervised models (PRE and CSUM) and the supervised data-driven ILUM model all have complementary advantages compared to each other, and they can be integrated together for building a more sophisticated hybrid user simulation model and it is a pivotal direction for future research of this thesis.

Additionally, different information-seeking theories including the economic model of IR [109] can also be integrated with the PRE model due to their similarity. The PRE model can be understood as a cost-gain framework of user model similar to the economic model of IR [109] where each decision costs time or effort of the user and results in relevant information, which is the gain of the user. The CSUM model introduces multiple heuristics regarding how the user IN may evolve and different components in the information need. We can also apply multiple information-seeking theories of users in this context to explore different theories and heuristics to model user cognitive states.

The precision, recall and effort modeling in the PRE framework can be instantiated with more complex models including supervised models. For example, precision, recall features and query length features can be combined through learning to rank and neural network models including a final layer aggregation of the three objectives such that the influence of the three objectives can be still interpretable. Thus the PRE model provides a general framework for developing simple to complex models including supervised models for query formulation. The CSUM model also has flexibility in that the IN updates can be improved as prior and posterior probability distribution updates by including the heuristics for updating the IN as part of the parameters of the probability distribution. Additionally, in both PRE and CSUM models, the user knowledge and IN updates are computed only using the word/unigram likelihood of the search engine results. These models can be improved by including n-grams or concepts/topics of the search results to better capture the learning process of the user. The ILUM model can also be improved by experimenting with different loss functions, deep learning frameworks, and action and task representations resulting in an effective user simulation model.

Finally, the user simulation models we propose are general frameworks that can be used to develop and study a wide range of simulation models from unsupervised to supervised models consisting of unigram language models to embedding-based representations. The CSUM model integrated with the PRE algorithm is more suitable for an E-commerce search or a structured search. The PRE model is more suitable to model web or text-based search whereas the CSUM model is better for E-commerce search but both models can be integrated and extended to better model both web and E-commerce search. The ILUM model is a supervised simulation model which can be applied to different search domains by using the respective search log data for training. The PRE and ILUM models can also be applied to literature searches. Therefore all the proposed models can be considered as fundamental frameworks that can be extended and improved to be applied to search simulation in different domains and the different models can also be integrated to better replicate real user search.

7.1 DISCUSSION

Limitations: Along with the multiple lines of extensions, as discussed above, we also describe the limitations of the proposed user simulation models and assumptions that may not always hold true and should be addressed depending on the application. One of the drawbacks of PRE and CSUM models is that probabilistic models for knowledge state, IN and their updates, which model background knowledge, preferences and learning process of the user, cannot be learnt or evaluated directly from real user data, as the user's knowledge and task are unobservable and thus cannot be evaluated or learned from. On the other hand, utilizing a supervised model like a deep learning network may learn to update the user action model from the context but it is uninterpretable and cannot be controlled to simulate different user behaviors. Second, in this thesis, we explore and evaluate different techniques for query simulation but we have studied only simplified methods for other user actions like click simulation, browsing and more. In addition, we assumed a simple linear user browsing behavior, where results are browsed in the ranking order. Many click simulation models based on different browsing theories have been proposed in previous works [14, 24, 26, 28, 40, 45, 59, 60, 61, 62], it is required to study these models by integrating them with PRE and CSUM models for improved click simulation and whether the query simulation and knowledge model affect the efficiency of click simulation. The click and browsing actions of a user can be influenced by different factors other than relevance. For example, every relevant document may not be the best choice for clicks, in the case of web search the trust and popularity of the webpages can be important variables affecting real user actions. The session-stopping action is also a complex user action which is less studied in the related work and in this thesis. Another limitation of the proposed models except the ILUM model is that they do not include a decision model to decide the next action of the user, i.e, we assume that the next action type like query or click is given and then generate the respective action using the model. It is required to model the above actions and decision models in order to apply the proposed models for different user simulation applications.

E-commerce search is more complex than web search due to multiple challenges like a larger set of actions, user preferences, decision making and different criteria for search ranking. In this thesis, we develop general frameworks that are applicable for both but it lacks in that it assumes a simplified E-commerce search behavior and does not consider some of the complexities. The search engine ranking may not be only based on relevance, different factors such as sponsored results, recommendations and product reviews that affect the ranking should be considered in the query formulation model and clicks actions. Additionally, the E-commerce search task is different from web search in that the user may be exploring very similar products (irrespective of relevance) and the decision to click and purchase can be affected by factors such as user preferences, consumer reviews, price and more. The above limitations should be addressed in order to obtain a more realistic E-commerce user simulation

Challenges in the simulation of online web and E-commerce users: Apart from the drawbacks of the proposed models, there are more open challenges for simulating online search users specifically E-commerce users and limitations in applying simulation models to replace real users.

The user models we propose in this thesis all assume certain simplifications to user browsing and search process, in order to formalize and derive computational models capturing/imitating user actions. However, most web and retail websites provide rich UX (user interface) to the users, and thus online search users have a complex browsing behavior performing multiple actions. Multiple different factors affect their browsing behavior. In the following, we discuss some strategies for extending the thesis work to relax some of the simplification assumptions in the current user models, so that the user models are better applicable for practical use and can be applied to simulate online users. However, note that capturing all of the complex user behavior could be an unsolvable open challenge.

One of the challenges to simulating online E-commerce search users is that most E-commerce websites provide rich UX for the user providing different options that help the searching or purchasing. For example, search interfaces have *filters* that allow filtering the ranked list of products based on ranges or sets of different attributes. Thus, while simulating online users we can incorporate simulating the action of selecting filters along with querying, clicking and other actions. The filter selection action can be simulated based on the product need. Another example is that the user click actions are influenced by different factors apart from relevance. For example, E-commerce websites provide the popularity of a product in terms of reviews and ratings which influence users' clicks, thus the product rating and popularity should also be included in the click simulation model of E-com users.

Another challenge is that E-commerce and web search engines not only rank the product according to relevance to the query but also according to many different factors, for example, sponsored results, customer reviews and so on. Thus the assumption that the results are ranked according to only relevance may not hold. Some of the methods to consider the above factors are to include the popularity of the product or Trustrank of webpages as part of the user simulation models. The notion of relevance is also different from the perspective of an E-commerce user, as many factors such as price, product item quantity, and the importance of different preferences play a role when deciding to purchase a product.

The assumption of ideal search engine results and the expert real user behavior may not always hold. In some cases, the search engine may fail to retrieve any relevant results. Real user behavior can also be very noisy and can be polluted by malicious users [134]. From the perspective of user simulation, such users may add noise to the search log, raising challenges in learning user simulation models. Interpretable simulation models may help alleviate this problem as they rely less on purely fitting the user data. **User tasks:** Most applications of simulation models like offline evaluation or training interactive models require large-scale session simulation. Therefore, it is likely required to automatically generate or sample a large number of information needs (INs) or user tasks to simulate a large number of search sessions for those INs.

A challenge of simulating E-commerce search is to generate tasks or product needs especially based on search log data. One of the problems is how to represent the user task, and the other problem is how to generate or sample a large number of tasks for simulating many sessions. The task representation can be simply a set of relevant documents or products, or it can be in the form of a textual description or questions or product preferences, or a more latent representation.

In the case of web search, most previous works utilize the benchmark datasets or curate topics(text description) or questions which serve as information needs. In some of the works, similar documents are grouped as a set of relevant documents representing the task, the task representation can be derived from the set. Therefore, a lot of information needs can be randomly sampled by using similarity between documents thus generating many search sessions.

In the case of E-commerce search, we can similarly sample a set of similar products as product need. However, the search log data can also be used to sample product need, for example, as done in the ILUM model where purchased products can be considered as product need. The product need can also be extracted from a search session in the form of preferred attributes or a more latent representation but latent representations cannot be controlled or changed meaningfully. Thus one of the challenges is how to best extract the user task information from a search session. In order to simulate a large number of sessions, the search log data can be used to obtain a distribution of product needs and sample the product needs according to the distribution. The above method can also be used for text-based search if search log data is available.

Limitations to replace real user evaluation or A/B testing: An important application of the user simulation model is to utilize them to interact with search systems for offline evaluation of the performance of the system. Currently, A/B testing is a commonly used method for evaluating and comparing search systems, however, this evaluation suffers from drawbacks such as it has the risk of exposing users to poorly performing systems, the method is also not reproducible, time-consuming and user intensive. Thus user simulation models can be used for offline A/B testing of the models. However, it is important to ensure the offline evaluation results will be consistent with the online results. Evaluating user simulation models for their ability to rank search systems alleviates this challenge to some extent in that we can estimate confidence in the offline evaluation of simulation models. Some of the previous works [14] and the RATE framework we proposed in this thesis evaluate a user simulation model by its ability to rank the IR systems correctly according to their performance. However, it is still challenging to confirm the result of the user simulator when comparing a given new search system with a previous system (offline A/B testing). Evaluation of the simulation model using different evaluation metrics can help provide more confidence in the offline evaluation results. Another solution is to specifically test if a given new search system performs poorly on any of the different product categories or topics before moving to online A/B testing.

7.2 FUTURE WORK

Integration of simulation models towards a unified user model: The simulation models proposed in this thesis each model the users in different ways. The simulation models each focus on modeling different aspects of the user search process in detail. An important research direction is to work towards a unified user model that is an integrated version of the multiple approaches and simulates all aspects of the user search process. For example, an agent framework for the user simulation that learns from real user-generated data, where it also models the cognitive state of the user during a search session which changes over time. The cognitive state may model both the knowledge and information need of the user, and it can be modelled as a Markov decision process where each state changes with the interactions from the search engine. Such a framework should also have input parameters representing user search behavior, knowledge and decision strategy during the search so that the simulator can be controlled to simulate different types of search sessions. For example, parameters controlling the knowledge level of the user, the exploratory behavior, patience, and effort/time spent by the user on search can be used as input parameters to the user simulation model to generate or output a user search behavior reflecting those parameters. Cognitive science theories and human-computer interaction theories can also be used in the development of user simulation models. Limitations of the proposed models and the challenges described earlier should be also addressed for a better user simulation model.

An ideal unified user model would include all the components described above. As another solution, multiple user simulator models can be developed as studied in this thesis and in practice, different user simulators can be applied together as an ensemble model either to evaluate an IIR system or for other applications of using simulation.

However, modeling all the aspects of the real user search behavior as a unified user model such that the simulated sessions are realistic may be a difficult challenge because the real user's cognitive process and decision process is unknown, ill-defined, affected by multiple factors and sometimes irrational.

Simulation based Evaluation platform: As discussed in this work, simulation has many applications, one of the challenges is to make user simulators easily accessible to the research community on a standard platform. We envision developing an evaluation platform with many user simulators available for the research community for utilizing in the evaluation of IR systems or other applications.

TREC tracks provide static data collections for the evaluation of IR systems, in contrast, our evaluation platform will consist of user simulators that simulate user actions and create a dynamic collection as it interacts with an IR system for evaluating it. This evaluation platform would consist of the collection of information needs, information object collection (documents or product items or more) to retrieve from, and multiple user simulators. An IR system can be evaluated with a selected user simulator, on the set of information needs and information object collection, the user simulator should select an information need and generate user actions relevant to it and interact with the IR system. Finally, the interaction is evaluated to compute a performance measure for the IR system. Further, the same setup can also be used for creating synthetic data using user simulators to train IIR systems. The platform is going to be flexible in that it can accommodate user simulators that can generate different types/sets of user actions relevant for evaluation in different domains. As a first step, a benchmark set of user simulators that can evaluate a standard IR search system or a standard conversational system can be included.

We propose to build this evaluation platform and make this available as a resource for evaluation in potentially two ways: 1)to make a simulator toolkit, where one has to configure it to use it for evaluation 2)as an evaluation web service, where one can choose a simulator and request for the simulator actions through API requests and then send responses of the respective IR systems that they want to evaluate. To simplify, the user simulator and the IR system will be interacting through an API.

Our proposed user models can also be part of this platform as one of the user simulators and the RATE framework which can be used to inform about the simulators on their reliability.

Different evaluation techniques to evaluate a user simulator: A simulator can be evaluated in three different ways similar to different types of validity concepts (replicative, predictive and structural) utilized to compare the simulation model to the original model [135].

First is whether the simulator can reproduce data that is similar to the real model; this is similar to replicative validity. Second, is whether the simulator can predict the real model's output for a given situation which is similar to predictive validity. The third is whether the simulation model is similar to the real user model in terms of its structure and function, i.e., whether the simulation model is similar to how the real user performs a search in terms of the action processes or decision processes. The third type, which is structural validity, is challenging or impossible to evaluate as we have little knowledge of the real user model or search process, and it is unobservable and hard to describe formally. One of the approximations for this type of evaluation is to perform user studies to identify the user strategies and compare them to the simulation model. The RATE framework proposed in this thesis is similar to predictive validity because RATE evaluates the simulator by using it to predict the result of comparing two IR systems if it is done by real users and compute the reliability based on how accurate the prediction is. The first type of evaluation which is replicative validity is utilized in PRE, CSUM and ILUM models and is also a common approach in previous literature. However, comparing simulated data to real user data still has many challenges. The similarity or equality of the simulated and actual actions studied in previous works only measure when both the actions are of the same action type and fail for different action types. Additionally, the evaluation metrics do not compare the sequence of actions. An important future work is to study and propose measurements for comparing different action types (like comparing query and click or click and purchase) generated by the simulator and the real user given the responses of the search engine. One of the measures, if a large amount of user-generated data is available, is to compare the generated action distribution of the simulator and the actual action distribution given the responses of the search engine.

REFERENCES

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[2] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, "Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer," in *Proceedings of the 28th ACM international conference on information and knowledge management*, 2019, pp. 1441–1450.

[3] D. Jannach, A. Manzoor, W. Cai, and L. Chen, "A survey on conversational recommender systems," *ACM Computing Surveys (CSUR)*, vol. 54, no. 5, pp. 1–36, 2021.

[4] Y. K. Dwivedi, N. Kshetri, L. Hughes, E. L. Slade, A. Jeyaraj, A. K. Kar, A. M. Baabdullah, A. Koohang, V. Raghavan, M. Ahuja et al., ""so what if chatgpt wrote it?" multidisciplinary perspectives on opportunities, challenges and implications of generative conversational ai for research, practice and policy," *International Journal of Information Management*, vol. 71, p. 102642, 2023.

[5] M. Sanderson, *Test collection based evaluation of information retrieval systems.* Now Publishers Inc, 2010.

[6] D. Kelly, *Methods for evaluating interactive information retrieval systems with users.* Now Publishers Inc, 2009.

[7] K. Hofmann, L. Li, and F. Radlinski, "Online evaluation for information retrieval," *Foundations and trends in information retrieval*, vol. 10, no. 1, pp. 1–117, 2016.

[8] J. Tague, M. Nelson, and H. Wu, "Problems in the simulation of bibliographic retrieval systems," in *Proceedings of the 3rd annual ACM conference on Research and development in information retrieval*, 1980, pp. 236–255.

[9] M. D. Cooper, "A simulation model of an information retrieval system," *Information Storage and Retrieval*, vol. 9, no. 1, pp. 13–32, 1973.

[10] C. Cleverdon, J. Mills, and M. Keen, "Factors determining the performance of indexing systems volume 1. design," 1966.

[11] D. Harman, "Overview of the rst trec conference," in *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1993, pp. 36–47.

[12] L. Azzopardi, K. Järvelin, J. Kamps, and M. D. Smucker, "Report on the sigir 2010 workshop on the simulation of interaction," *SIGIR Forum*, vol. 44, no. 2, pp. 35–47, Jan. 2011.

[13] Y. Zhang, X. Liu, and C. Zhai, "Information retrieval evaluation as search simulation: A general formal framework for ir evaluation," in *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval.* ACM, 2017, pp. 193–200.

[14] B. Carterette, A. Bah, and M. Zengin, "Dynamic test collections for retrieval evaluation," in *Proceedings of the 2015 international conference on the theory of information retrieval.* ACM, 2015, pp. 91–100.

[15] A. Lipani, B. Carterette, and E. Yilmaz, "How am i doing?: Evaluating conversational search systems offline," *ACM Transactions on Information Systems (TOIS)*, vol. 39, no. 4, pp. 1–22, 2021.

[16] J. Schatzmann, B. Thomson, K. Weilhammer, H. Ye, and S. Young, "Agenda-based user simulation for bootstrapping a POMDP dialogue system," in *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, ser. NAACL-HLT '07, 2007, pp. 149–152.

[17] W. Zhang, X. Zhao, L. Zhao, D. Yin, and G. H. Yang, "Drl4ir: 2nd workshop on deep reinforcement learning for information retrieval," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 2681–2684.

[18] X. Chen, L. Yao, J. McAuley, G. Zhou, and X. Wang, "A survey of deep reinforcement learning in recommender systems: A systematic review and future directions," *arXiv preprint arXiv:2109.03540*, 2021.

[19] F. Baskaya, H. Keskustalo, and K. Järvelin, "Modeling behavioral factors in interactive information retrieval," in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management.* ACM, 2013, pp. 2297–2302.

[20] L. Azzopardi and G. Zuccon, "An analysis of the cost and benefit of search interactions," in *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, ser. ICTIR '16, 2016, pp. 59–68.

[21] D. Bountouridis, J. Harambam, M. Makhortykh, M. Marrero, N. Tintarev, and C. Hauff, "Siren: A simulation framework for understanding the effects of recommender systems in online news environments," in *Proceedings of the Conference on Fairness, Accountability, and Transparency*, ser. FAT* '19, 2019, pp. 150–159.

[22] C. Jordan, C. Watters, and Q. Gao, "Using controlled query generation to evaluate blind relevance feedback algorithms," in *Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries.* ACM, 2006, pp. 286–295.

[23] S. Verberne, M. Sappelli, K. Järvelin, and W. Kraaij, “User simulations for interactive search: Evaluating personalized query suggestion,” in *European Conference on Information Retrieval.* Springer, 2015, pp. 678–690.

[24] R. W. White, “Using searcher simulations to redesign a polyrepresentative implicit feedback interface,” *Inf. Process. Manage.*, vol. 42, no. 5, pp. 1185–1202, Sep. 2006.

[25] H. Keskustalo, K. Järvelin, and A. Pirkola, “Evaluating the effectiveness of relevance feedback based on a user simulation model: Effects of a user scenario on cumulated gain value,” *Inf. Retr.*, vol. 11, no. 3, pp. 209–228, June 2008.

[26] K. Järvelin, “Interactive relevance feedback with graded relevance and sentence extraction: Simulated user experiments,” in *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, ser. CIKM ’09, 2009, pp. 2053–2056.

[27] R. Jagerman, H. Oosterhuis, and M. de Rijke, “To model or to intervene: A comparison of counterfactual and online learning to rank from user interactions,” in *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*, 2019, pp. 15–24.

[28] M. Zoghi, T. Tunys, M. Ghavamzadeh, B. Kveton, C. Szepesvari, and Z. Wen, “Online learning to rank in stochastic click models,” in *International Conference on Machine Learning.* PMLR, 2017, pp. 4199–4208.

[29] J. McInerney, E. Elahi, J. Basilico, Y. Raimond, and T. Jebara, “Accordion: A trainable simulator for long-term interactive systems,” in *Fifteenth ACM Conference on Recommender Systems*, 2021, pp. 102–113.

[30] D. Griol, J. Carbó, and J. M. Molina, “An automatic dialog simulation technique to develop and evaluate interactive conversational agents,” *Appl. Artif. Intell.*, vol. 27, no. 9, pp. 759–780, oct 2013.

[31] S. Young, M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu, “The hidden information state model: A practical framework for pomdp-based spoken dialogue management,” *Computer Speech & Language*, vol. 24, no. 2, pp. 150–174, 2010.

[32] A. Salle, S. Malmasi, O. Rokhlenko, and E. Agichtein, “Studying the effectiveness of conversational search refinement through user simulation,” in *European Conference on Information Retrieval.* Springer, 2021, pp. 587–602.

[33] S. Zhang and K. Balog, “Evaluating conversational recommender systems via user simulation,” in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1512–1520.

[34] S. Labhishetty, C. Zhai, S. Ranganath, and P. Ranganathan, “A cognitive user model for e-commerce search,” in *Proceedings of the Data Science for Retail and E-Commerce Workshop*, 2020.

[35] S. Labhishetty and C. Zhai, "Pre: A precision-recall-effort optimization framework for query simulation," in *Proceedings of the 2022 ACM SIGIR International Conference on Theory of Information Retrieval*, 2022, pp. 51–60.

[36] S. Labhishetty and C. Zhai, "An exploration of tester-based evaluation of user simulators for comparing interactive retrieval systems." in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3404835.3463091 p. 1598–1602.

[37] A. Muhamed, I. Keivanloo, S. Perera, J. Mracek, Y. Xu, Q. Cui, S. Rajagopalan, B. Zeng, and T. Chilimbi, "Ctr-bert: Cost-effective knowledge distillation for billion-parameter teacher models," in *NeurIPS Efficient Natural Language and Speech Processing Workshop*, 2021.

[38] H. Yang, P. Gupta, R. F. Galan, D. Bu, and D. Jia, "Seasonal relevance in e-commerce search," in *CIKM 2021*, 2021. [Online]. Available: https://www.amazon.science/publications/seasonal-relevance-in-e-commerce-search

[39] D. Maxwell and L. Azzopardi, "Agents, simulated users and humans: An analysis of performance and behaviour," in *Proceedings of the 25th ACM international on conference on information and knowledge management*. ACM, 2016, pp. 731–740.

[40] F. Baskaya, H. Keskustalo, and K. Jarvelin, "Simulating simple and fallible relevance feedback," in *Proceedings of ECIR*, 2011.

[41] F. Baskaya, "Simulating search sessions in interactive information retrieval evaluation," 2014, phD thesis, University of Tempere.

[42] H. Keskustalo, K. Järvelin, A. Pirkola, T. Sharma, and M. Lykke, "Test collection-based ir evaluation needs extension toward sessions–a case of extremely short queries," in *Asia Information Retrieval Symposium*. Springer, 2009, pp. 63–74.

[43] L. Azzopardi, M. De Rijke, and K. Balog, "Building simulated queries for known-item topics: an analysis using six european languages," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2007, pp. 455–462.

[44] D. Maxwell, L. Azzopardi, K. Järvelin, and H. Keskustalo, "Searching and stopping: An analysis of stopping rules and strategies," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, ser. CIKM '15. New York, NY, USA: ACM, 2015. [Online]. Available: http://doi.acm.org/10.1145/2806416.2806476 pp. 313–322.

[45] A. Chuklin, I. Markov, and M. d. Rijke, "Click models for web search," *Synthesis Lectures on Information Concepts, Retrieval, and Services*, vol. 7, no. 3, pp. 1–115, 2015.

[46] F. Baskaya, H. Keskustalo, and K. Järvelin, "Time drives interaction: Simulating sessions in diverse searching environments," in *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, 2012, pp. 105–114.

[47] N. J. Belkin, R. N. Oddy, and H. M. Brooks, "Ask for information retrieval: Part i. background and theory," *Journal of documentation*, vol. 38, no. 2, pp. 61–71, 1982.

[48] P. Ingwersen, "Cognitive perspectives of information retrieval interaction: elements of a cognitive ir theory," *Journal of documentation*, vol. 52, no. 1, pp. 3–50, 1996.

[49] K. Balog, D. Maxwell, P. Thomas, and S. Zhang, "Report on the 1st simulation for information retrieval workshop (sim4ir 2021) at sigir 2021," in *ACM SIGIR Forum*, vol. 55, no. 2. ACM New York, NY, USA, 2022, pp. 1–16.

[50] R. P. Bhattacharyya, D. J. Phillips, B. Wulfe, J. Morton, A. Kuefler, and M. J. Kochenderfer, "Multi-agent imitation learning for driving simulation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1534–1539.

[51] E. Rohmer, S. P. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 1321–1326.

[52] F. Golemo, A. A. Taiga, A. Courville, and P.-Y. Oudeyer, "Sim-to-real transfer with neural-augmented robot simulation," in *Conference on Robot Learning*. PMLR, 2018, pp. 817–828.

[53] M. C. Fu, "Alphago and monte carlo tree search: the simulation optimization perspective," in *2016 Winter Simulation Conference (WSC)*. IEEE, 2016, pp. 659–670.

[54] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Comput. Surv.*, vol. 50, no. 2, apr 2017. [Online]. Available: https://doi.org/10.1145/3054912

[55] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 29, 2016.

[56] O. Pietquin, "Inverse reinforcement learning for interactive systems," in *Proceedings of the 2nd Workshop on Machine Learning for Interactive Systems: Bridging the Gap Between Perception, Action and Communication*, ser. MLIS '13. New York, NY, USA: Association for Computing Machinery, 2013. [Online]. Available: https://doi.org/10.1145/2493525.2493529 p. 71–75.

[57] S. Chandramohan, M. Geist, F. Lefevre, and O. Pietquin, "User simulation in dialogue systems using inverse reinforcement learning," in *Twelfth annual conference of the international speech communication association*, 2011.

[58] Z. Li, J. Kiseleva, and M. De Rijke, "Dialogue generation: From imitation learning to inverse reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 6722–6729.

[59] B. Carterette, E. Kanoulas, and E. Yilmaz, "Simulating simple user behavior for system effectiveness evaluation," in *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, ser. CIKM '11, 2011, pp. 611–620.

[60] J. Zhang, J. Mao, Y. Liu, R. Zhang, M. Zhang, S. Ma, J. Xu, and Q. Tian, "Context-aware ranking by constructing a virtual environment for reinforcement learning," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 1603–1612.

[61] R. W. White, I. Ruthven, J. M. Jose, and C. J. V. Rijsbergen, "Evaluating implicit feedback models using searcher simulations," *ACM Trans. Inf. Syst.*, vol. 23, no. 3, pp. 325–361, July 2005.

[62] H. Keskustalo, K. Järvelin, and A. Pirkola, "The effects of relevance feedback quality and quantity in interactive relevance feedback: A simulation based on user modeling," in *Proceedings of the 28th European Conference on Advances in Information Retrieval*, ser. ECIR '06, 2006, pp. 191–204.

[63] D. Maxwell and L. Azzopardi, "Simulating interactive information retrieval: Simiir: A framework for the simulation of interaction," in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2016, pp. 1141–1144.

[64] D. Maxwell, L. Azzopardi, and Y. Moshfeghi, "A study of snippet length and informativeness: Behaviour, performance and user experience," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '17, 2017, pp. 135–144.

[65] D. Maxwell and L. Azzopardi, "Information scent, searching and stopping: Modelling serp level stopping behaviour," in *Proceedings of the 40th European Conference on IR Research*, ser. ECIR '18, 2018, pp. 210–222.

[66] P. Erbacher, L. Soulier, and L. Denoyer, "State of the art of user simulation approaches for conversational information retrieval," *arXiv preprint arXiv:2201.03435*, 2022.

[67] J. Schatzmann, K. Weilhammer, M. Stuttle, and S. Young, "A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies," *Knowl. Eng. Rev.*, vol. 21, no. 2, pp. 97–126, June 2006.

[68] W. Shi, K. Qian, X. Wang, and Z. Yu, "How to build user simulators to train rl-based dialog systems," *arXiv preprint arXiv:1909.01388*, 2019.

[69] Q. Zhu, Z. Zhang, Y. Fang, X. Li, R. Takanobu, J. Li, B. Peng, J. Gao, X. Zhu, and M. Huang, "Convlab-2: An open-source toolkit for building, evaluating, and diagnosing dialogue systems," *arXiv preprint arXiv:2002.04793*, 2020.

[70] J. Schatzmann, B. Thomson, K. Weilhammer, H. Ye, and S. Young, "Agenda-based user simulation for bootstrapping a pomdp dialogue system," in *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, 2007, pp. 149–152.

[71] J. Afzali, A. M. Drzewiecki, K. Balog, and S. Zhang, "Usersimcrs: A user simulation toolkit for evaluating conversational recommender systems," *arXiv preprint arXiv:2301.05544*, 2023.

[72] P. Sondhi, M. Sharma, P. Kolari, and C. Zhai, "A taxonomy of queries for e-commerce search," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018, pp. 1245–1248.

[73] S. Labhishetty, C. Zhai, M. Xie, L. Gong, R. Sharnagat, and S. Chembolu, "Differential query semantic analysis: Discovery of explicit interpretable knowledge from e-com search logs," in *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, 2022, pp. 535–543.

[74] M. Mao, J. Lu, J. Han, and G. Zhang, "Multiobjective e-commerce recommendations based on hypergraph ranking," *Information Sciences*, vol. 471, pp. 269–287, 2019.

[75] S. Liu, F. Xiao, W. Ou, and L. Si, "Cascade ranking for operational e-commerce search," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 1557–1565.

[76] Q. Guo and E. Agichtein, "Ready to buy or just browsing? detecting web searcher goals from interaction data," in *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, 2010, pp. 130–137.

[77] A. Hassan, R. W. White, S. T. Dumais, and Y.-M. Wang, "Struggling or exploring?: disambiguating long search sessions," in *Proceedings of the 7th ACM international conference on Web search and data mining.* ACM, 2014, pp. 53–62.

[78] A. Hassan Awadallah, R. W. White, P. Pantel, S. T. Dumais, and Y.-M. Wang, "Supporting complex search tasks," in *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*, 2014, pp. 829–838.

[79] K. Wang, N. Gloy, and X. Li, "Inferring search behaviors using partially observable markov (pom) model," in *Proceedings of the third ACM international conference on Web search and data mining*, 2010, pp. 211–220.

[80] M. D. Smucker and C. L. Clarke, "Modeling optimal switching behavior," in *Proceedings of the 2016 ACM on Conference on Human Information Interaction and Retrieval*, ser. CHIIR '16, 2016, pp. 317–320.

[81] T. Pääkkönen, K. Järvelin, J. Kekäläinen, H. Keskustalo, F. Baskaya, D. Maxwell, and L. Azzopardi, "Exploring behavioral dimensions in session effectiveness," in *Proceedings of the 6th International Conference on Experimental IR Meets Multilinguality, Multimodality, and Interaction*, ser. CLEF '15, 2015, pp. 178–189.

[82] L. Azzopardi, "The economics in interactive information retrieval," in *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '11, 2011, pp. 15–24.

[83] L. Azzopardi, D. Kelly, and K. Brennan, "How query cost affects search behavior," in *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '13, 2013, pp. 23–32.

[84] Y. He, J. Tang, H. Ouyang, C. Kang, D. Yin, and Y. Chang, "Learning to rewrite queries," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, 2016, pp. 1443–1452.

[85] A. Herdagdelen, M. Ciaramita, D. Mahler, M. Holmqvist, K. Hall, S. Riezler, and E. Alfonseca, "Generalized syntactic and semantic models of query reformulation," in *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, 2010, pp. 283–290.

[86] R. Jones, B. Rey, O. Madani, and W. Greiner, "Generating query substitutions," in *Proceedings of the 15th international conference on World Wide Web*, 2006, pp. 387–396.

[87] I. Ruthven, "Re-examining the potential effectiveness of interactive query expansion," in *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, ser. SIGIR '03, 2003, pp. 213–220.

[88] L. Azzopardi and M. de Rijke, "Automatic construction of known-item finding test beds," in *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '06, pp. 603–604.

[89] S. Labhishetty and C. Zhai, "Rate: A reliability-aware tester-based evaluation framework of user simulators," in *European Conference on Information Retrieval.* Springer, 2022, pp. 336–350.

[90] T. Breuer, N. Fuhr, and P. Schaer, "Validating simulations of user query variants," in *Advances in Information Retrieval*, M. Hagen, S. Verberne, C. Macdonald, C. Seifert, K. Balog, K. Nørvåg, and V. Setty, Eds. Cham: Springer International Publishing, 2022, pp. 80–94.

[91] A. Grotov, A. Chuklin, I. Markov, L. Stout, F. Xumara, and M. de Rijke, "A comparative study of click models for web search," in *International Conference of the Cross-Language Evaluation Forum for European Languages.* Springer, 2015, pp. 78–90.

[92] A. Borisov, I. Markov, M. De Rijke, and P. Serdyukov, "A neural click model for web search," in *Proceedings of the 25th International Conference on World Wide Web*, 2016, pp. 531–541.

[93] F. Guo, C. Liu, and Y. M. Wang, "Efficient multiple-click models in web search," in *Proceedings of the second acm international conference on web search and data mining*, 2009, pp. 124–131.

[94] A. Câmara, D. Maxwell, and C. Hauff, "Searching, learning, and subtopic ordering: A simulation-based analysis," *arXiv preprint arXiv:2201.11181*, 2022.

[95] L. Azzopardi, "Modelling interaction with economic models of search," in *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval.* ACM, 2014, pp. 3–12.

[96] T. Pääkkönen, J. Kekäläinen, H. Keskustalo, L. Azzopardi, D. Maxwell, and K. Järvelin, "Validating simulated interaction for retrieval evaluation," *Information Retrieval Journal*, vol. 20, no. 4, pp. 338–362, 2017.

[97] H. Yang, D. Guan, and S. Zhang, "The query change model: Modeling session search as a markov decision process," *ACM Transactions on Information Systems (TOIS)*, vol. 33, no. 4, pp. 1–33, 2015.

[98] D. Guan, S. Zhang, and H. Yang, "Utilizing query change for session search," in *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, 2013, pp. 453–462.

[99] C. Eickhoff, J. Teevan, R. White, and S. Dumais, "Lessons from the journey: a query log analysis of within-session learning," in *Proceedings of the 7th ACM international conference on Web search and data mining*, 2014, pp. 223–232.

[100] F. Moraes, S. R. Putra, and C. Hauff, "Contrasting search as a learning activity with instructor-designed learning," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 167–176.

[101] N. Bhattacharya and J. Gwizdka, "Relating eye-tracking measures with changes in knowledge on search tasks," in *Proceedings of the 2018 ACM symposium on eye tracking research & applications*, 2018, pp. 1–5.

[102] B. M. Wildemuth, "The effects of domain knowledge on search tactic formulation," *Journal of the american society for information science and technology*, vol. 55, no. 3, pp. 246–258, 2004.

[103] H. L. O'Brien, A. Kampen, A. W. Cole, and K. Brennan, "The role of domain knowledge in search as learning," in *Proceedings of the 2020 Conference on Human Information Interaction and Retrieval*, 2020, pp. 313–317.

[104] M. J. Bates, "The design of browsing and berrypicking techniques for the online search interface," *Online review*, 1989.

[105] P. Ingwersen, *Integrative framework for information seeking and interactive information retrieval.* na, 2005.

[106] C. C. Kuhlthau, "Developing a model of the library search process: Cognitive and affective aspects," *Rq*, pp. 232–242, 1988.

[107] P. Pirolli and S. Card, "Information foraging." *Psychological review*, vol. 106, no. 4, p. 643, 1999.

[108] D. Ellis, "Modeling the information-seeking patterns of academic researchers: A grounded theory approach," *The Library Quarterly*, vol. 63, no. 4, pp. 469–486, 1993.

[109] L. Azzopardi, "The economics in interactive information retrieval," in *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, 2011, pp. 15–24.

[110] B. Carterette, E. Kanoulas, M. Hall, and P. Clough, "Overview of the trec 2014 session track," DELAWARE UNIV NEWARK DEPT OF COMPUTER AND INFORMATION SCIENCES, Tech. Rep., 2014.

[111] C. Zhai, "Interactive information retrieval: Models, algorithms, and evaluation," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 2444–2447.

[112] S. I. Nikolenko, S. Koltcov, and O. Koltsova, "Topic modelling for qualitative studies," *Journal of Information Science*, vol. 43, no. 1, pp. 88–102, 2017.

[113] J. Chi, J. Ouyang, C. Li, X. Dong, X. Li, and X. Wang, "Topic representation: Finding more representative words in topic models," *Pattern recognition letters*, vol. 123, pp. 53–60, 2019.

[114] Y. Ni, Q. K. Xu, F. Cao, Y. Mass, D. Sheinwald, H. J. Zhu, and S. S. Cao, "Semantic documents relatedness using concept graph representation," in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, 2016, pp. 635–644.

[115] P. Norvig, "Natural language corpus data: Beautiful data," 2008. [Online]. Available: http://norvig.com/ngrams/

[116] C. Van Gysel, E. Kanoulas, and M. de Rijke, "Pyndri: a python interface to the indri search engine," in *ECIR*, vol. 2017. Springer, 2017.

[117] S. Niwattanakul, J. Singthongchai, E. Naenudorn, and S. Wanapu, "Using of jaccard coefficient for keywords similarity," in *Proceedings of the international multiconference of engineers and computer scientists*, vol. 1, no. 6, 2013, pp. 380–384.

[118] T. Sellam, D. Das, and A. P. Parikh, "Bleurt: Learning robust metrics for text generation," *arXiv preprint arXiv:2004.04696*, 2020.

[119] D. Wang, G. Wang, X. Ke, and W. Chen, "Action prediction and identification from mining temporal user behaviors," in *Proceedings of the fourth ACM international conference on Web search and data mining*, 2011, pp. 435–444.

[120] S. Malkevich, I. Markov, E. Michailova, and M. De Rijke, "Evaluating and analyzing click simulation in web search," in *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, 2017, pp. 281–284.

[121] N. Su, J. He, Y. Liu, M. Zhang, and S. Ma, "User intent, behaviour, and perceived satisfaction in product search," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 2018, pp. 547–555.

[122] M. Bain and C. Sammut, "A framework for behavioural cloning." in *Machine Intelligence 15*, 1995, pp. 103–129.

[123] T. K. Devi, A. Srivatsava, K. K. Mudgal, R. R. Jayanti, and T. Karthick, "Behaviour cloning for autonomous driving." *Webology*, vol. 17, no. 2, pp. 694–705, 2020.

[124] S. Arora and P. Doshi, "A survey of inverse reinforcement learning: Challenges, methods and progress," *Artificial Intelligence*, vol. 297, p. 103500, 2021.

[125] D. Maxwell, L. Azzopardi, K. Järvelin, and H. Keskustalo, "An initial investigation into fixed and adaptive stopping strategies," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2015, pp. 903–906.

[126] F. Baskaya, H. Keskustalo, and K. Jarvelin, "Time drives interaction: simulating sessions in diverse searching environments," in *Proceedings of ACM SIGIR*, 2012.

[127] B. P. Zeigler, T. G. Kim, and H. Praehofer, *Theory of Modeling and Simulation*, 2nd ed. USA: Academic Press, Inc., 2000.

[128] J. M. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. S. Tomkins, "The web as a graph: Measurements, models, and methods," in *International Computing and Combinatorics Conference*. Springer, 1999, pp. 1–17.

[129] S. Sriram, X. Shen, and C. Zhai, "A session-based search engine," in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, 2004, pp. 492–493.

[130] J. Rocchio, "Relevance feedback in information retrieval," *The Smart retrieval system-experiments in automatic document processing*, pp. 313–323, 1971.

[131] A. Trotman, A. Puurula, and B. Burgess, "Improvements to bm25 and language models examined," in *Proceedings of the 2014 Australasian Document Computing Symposium*, 2014, pp. 58–65.

[132] M. Chaput, "Whoosh," https://whoosh.readthedocs.io/en/latest/#.

[133] J. Jiang and J. Allan, "Correlation between system and user metrics in a session," in *Proceedings of the 2016 ACM on Conference on Human Information Interaction and Retrieval*, 2016, pp. 285–288.

[134] N. Jagpal, E. Dingle, J.-P. Gravel, P. Mavrommatis, N. Provos, M. A. Rajab, and K. Thomas, "Trends and lessons from three years fighting malicious extensions," in *24th {USENIX} Security Symposium ({USENIX} Security 15)*, 2015, pp. 579–593.

[135] B. P. Zeigler, T. G. Kim, and H. Praehofer, *Theory of modeling and simulation*. Academic press, 2000.