Beyond Query: Interactive User Intention Understanding

Yang Yang
Department of Computer Science and Technology
Tsinghua University
Beijing, China
SherlockBourne@gmail.com

Jie Tang
Department of Computer Science and Technology
Tsinghua University
Beijing, China
jietang@tsinghua.edu.cn

***Abstract*—Despite the powerful search engine technologies, users often fail to capture one or more words as queries to describe their target information, and fail to find the information they are seeking for. Techniques such as user intention predictions and personalized recommendations are designed so as to help the users figure out how to reach their target. In this work, we aim to help users identify their target items using a new approach called *Interactive User Intention Understanding*. In particular, we construct an automatic questioner that generates yes-or-no questions for the user in each round. Then we infer the user intention according to the corresponding answers. In order to generate "smart" questions in optimal sequence, we propose the *IHS* algorithm based on heuristic search. In addition, we theoretically prove a bound on the ranking of target items given the questions and answers. We implement our experiments on three data sets and compare our result with two baseline methods. Experimental results show that *IHS* outperforms the baseline methods by 27.83% and 25.98% respectively.**

I. Introduction

Despite the exponential growth of information and powerful search engine technologies, we often fail to find the exact information that we want. Studies show that approximately 28% of roughly 2 billion daily web searches are modifications to a previous query [18]. In addition, when users attempt to fill a s ingle information need, 52% of them will modify their queries [11]. Among those modifications, 35.2% are totally changing the query while only 7.1% are simply adding terms [21]. Statistical results tell that users' needs are sometimes too vague to be stated clearly.

Psychologists and educators believe that question-and-answering convey more precise information than mere statements [22]. This inspires us with an interactive question-and-answer approach to help users identify their needs. Specifically, we aim to construct an automatic questioner, that generates questions for the user in each round. We then infer the user intention according to the corresponding answers. More formally, by assuming a hypothetical space which contains the user's target, we partition the hypothetical space according to the user's response to the questions and let it finally converge to the user's target.

This method, namely *interactive user intention understanding*, is a novel issue and lacks comprehensive study. The major challenges include:

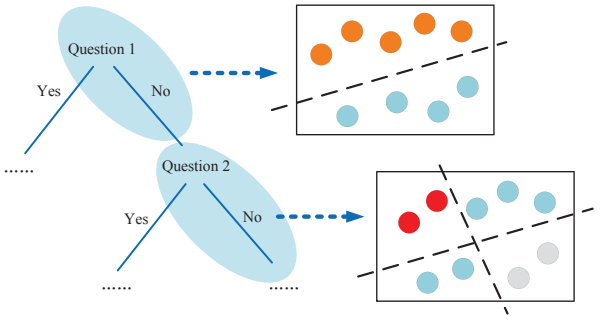


Figure 1. An example of the hypothesis space being partitioned by two questions. The left part stands for a decision tree constituted by questions. Two boxes on the right part stand for two statuses of the same hypothesis space. Bright points have higher possibilities to be the target than pale points.

Question generation. Good questions should contributes to partition the hypothesis space. Also, a good question sequence should identify the target within fewer number of rounds. Therefore a core challenge of this work is how to generate the optimal sequence of questions.

User response. There are typically two types of answers that we can design: literal statements or option choices. Literal statements convey more information, at the cost of efficiency – they require time both for users to type them in and for system to process them. Option choices are more user-friendly but provide limited information. Thus another challenging issue is how to design proper answer types to balance the trade-off between these methods.

Interaction efficiency. The time cost of each interactive round includes: (1) time used for the questioner to generate questions, (2) time used for user to response, and (3) time cost for updating the hypothesis space. In real applications the hypothesis space will be very large. How to design efficient algorithms to deal with large-scale data and make the interaction prompt is the third challenge.

We address all three challenges in our approach, which only requires users to answer in binary forms: "yes" or "no". Figure 1 shows an example of the interaction process. Question 1 could be "Are you looking for a Chinese restaurant?" and we suppose the user answers "no". Then

the points standing for Chinese restaurants (lower part) will be assigned with lower probabilities. Similar procedure is repeated for Question 2. We then obtain four regions with three different probability values: red points that are more likely to be the target and grey points that are less likely.

In this paper, we make three contributions:

First, we propose an interactive method that does not require typing. Simplified operation is meaningful especially for mobile users, considering the inefficiency of mobile typing as well as the rapid growth of Mobile Internet.

Second, we model the user's probability of making mistakes and prove a bound of target ranking. Concretely, we realize that during the interactive process, the user may answer the questions incorrectly by mis-operations. We assume each user make mistakes with some probability, and we bound the ranking of the target when we have asked some number of questions. The bound also depends on the probability of "mistaken answers".

Third, we design an Interactive Heuristic Search (IHS) method to generate sequential questions and partition the hypothesis space according to the user's answers. We compare the proposed algorithm with baseline approaches on three data sets. On average, IHS requires 7.47 questions less than others to identify the target. Experiments show that ignoring user's response time, IHS only takes 0.16 second on average to finish an interactive round.

The paper is organized as follows. Section II reviews previous related works. Section III formulates the problem. Section IV introduces our proposed algorithm. Section V gives the theoretical basis for the proposed algorithm. Section VI describes the experiments we conduct to validate the effectiveness of our methodology. Section VII concludes this work.

II. Related Work

Intention Prediction. Predicting user's intention based on user's query or other information is becoming a challenging task. Bauer [2] introduced some typical methods that train agents to identify and extract interesting pieces of information from online documents. Fragoudis and Likothanassis introduced the retriever, an autonomous agent that executes user-queries and returns high quality results in [7]. Such agent makes use of existing search engines and conducts self-training in order to analyze the user's preference from semantic information.

Another approach developed by Chen [4] models and infers user's actions. This approach not only considers keyword features, but also tries to form a concept hierarchy of keywords to improve the performance. Other information agents like Office Assistant [9] and WebWatcher [1] use similar approaches.

Recommendation and Link Analysis. To understand users' intentions, various issues based on search engines and recommendation systems were developed. Link analysis [17], [12], [14] is a data-analysis technique used to evaluate relationships between nodes in the network. Techniques like this will help the system find out the popular items, which will in turn help users search for valuable items and information. Many recommendation engines were also developed. Typically, users need to register and tell the recommendation engines their preferences on some items explicitly, since most of traditional recommendation methods rely heavily on user logs and feedbacks [13]. In [19], authors make analysis on different item-based recommendation generation algorithm, including the computation of similarities between items and the way of obtaining recommendations. Meanwhile, they made comparisons between their results and basic k-nearest neighbor approach [6], [5], which is based on collaborative filtering and is a popular recommendation system.

However, recommendation engines may not produce meaningful recommendations when users can not express their preferences accurately or there is no available user logs.

Others As we mentioned before, the core challenge of our work is the question selection problem, which is similar with active learning problems [20], [25]. However, we model the probability of users making mistakes in our work, which makes it different from active learning. Also, partitioning algorithms such as KD tree [16], [10] and learning algorithms such as neural networks [3] can be applied to solve the question selection problem. Another widely used algorithm is the greedy, which will be briefly introduced in section IV as a baseline.

The system involving the problem of finding target item by asking yes-or-no questions have similar patterns. Usually the system asks the player a yes-or-no question, and the player answers it. Getting the answer of the player, the system use some algorithm to find next question and present it to the player. The selection of the question aim to minimize the total number of questions to be asked to identify the target item.

It is usually very useful to analyze the number of questions to be asked such that the target item is ranked in top n. Doing such analysis requires considering the worst case in the procedure, in this game it's the "worst answer" that the player could possibly give. When we consider this game in the worst case, it becomes similar with another famous game called "Pusher-Chooser" introduced by Spencer [23], in which pusher has to find a balanced split whereas chooser tries to take the advantage to imbalance.

III. Problem

In this section, we introduce some necessary definitions as well as the formulation of the problem. Suppose we have a set of n items as candidates for users to choose from, which are embedded in an m-dimensional tag space. Each tag can be viewed as an attribute or a feature of items.

Definition 1: **Tag Matrix.** A tag matrix $\mathbf{X}$ is an $n \times m$ matrix that indicates the relationship between items and tags, where $x_{ij} = 1$ if and only if item i contains tag j.

Definition 2: **Yes-or-No Question.** A yes-or-no question can be regarded as a two-tuple (q_i, l), where q_i is a tag used in the i-th question and $l \in \{0, 1\}$ stands for the user's answer. Here $l = 1$ denotes "yes" and $l = 0$ denotes "no".

As we mentioned before, we only use general questions that require user to answer "yes" or "no". The format for the i-th question is asking whether the *target item* the user is looking for contains tag q_i.

Problem & **System:** In this problem, we are given a tag matrix X, and our aim is to interactively understand the users' intentions in several iterations.

More precisely, we assume that there is a **latent target item** that best fits the users' intentions before the users interact with our system. In each iteration we present a yes-or-no question to the users, and according to the answers of the users we find next questions to be presented, and show a ranking list of items as the recommendations to the users. For example, a user is looking for some software company and the latent target item that best fits her intention is IBM. Then our system asks yes-or-no questions, and according to the answer of the user the system presents the ranking to the user.

Since there is no "official" way to evaluate the performance of such system on this problem up to our knowledge, we formulate the evaluation as the number of questions asked, i.e., our goal is to minimize the number of questions asked.

IV. Approach

We introduce our approach in this section. The idea is to assign each item a weight. In each question-and-answer round, we update item weights to ensure the items that "mismatch" the answer have smaller weight and vice versa, and then we present the ranking of items according to the weights to users.

A. Framework

We use a vector $\mathbf{w^{(t)}}$ to denote the weights of all items after updating in the t-th round. All weights of items are initialized to 1, namely $\mathbf{w^{(0)}} = \mathbf{1}$. We update the weights by discounting. Specifically, in the t-th round, for items that mismatch the answer, we multiply a *discount factor* γ ($0 \leq \gamma < 1$) to the weight $w_i^{(t)}$. For example, if the user gives negative answer to tag j, then every item i containing tag j will receive a discounted weight. We do this and present the ranking of items in each round.

Next we will introduce two algorithms for finding the tag used in each question.

Algorithm 1 The interactive recommendation framework.

Input: a tag matrix X, the maximum number of questions T, and the discount factor γ.
Output: an updated weight vector $\mathbf{w^{(T)}}$.

```
initialize weights w_i^(0) = 1 for all i
t = 0
repeat
    q = QuestionSelectionAlgo(X, w^t, γ)
    Ask the user if target item contains q
    if the answer is "yes" then
        w_i^(t+1) = w_i^(t) γ^(1−x_iq) for all i
    else
        w_i^(t+1) = w_i^(t) γ^(x_iq) for all i
    end if
    t = t + 1
until t = T
return w^(T)
```

B. Greedy

We first introduce a greedy algorithm as the baseline. The basic idea is to choose the tag that decreases $\left\|\mathbf{w}^{(t)}\right\|_1$ the most in the worst case, where $\mathbf{w}^{(t)}$ is the weight vector after t-th question-and-answer round. Let $w_i^{(t)}$ be the i-th entry of $\mathbf{w}^{(t)}$.

Assume that we ask whether the target item contains tag q. Items are divided into 2 groups by q: the one with tag q and the other one without tag q. It follows that the corresponding total weight for these 2 groups are $\langle \mathbf{w}^{(t)}, \mathbf{X}_q \rangle$ and $\langle \mathbf{w}^{(t)}, \mathbf{1} - \mathbf{X}_q \rangle$ respectively, where $\mathbf{X}_q$ is the q-th column of the tag matrix $\mathbf{X}$. Then we set α as follows:

$$\alpha = \min \left\{ \frac{\langle \mathbf{w}^{(t)}, \mathbf{X}_q \rangle}{\|\mathbf{w}^{(t)}\|_1}, \frac{\langle \mathbf{w}^{(t)}, \mathbf{1} - \mathbf{X}_q \rangle}{\|\mathbf{w}^{(t)}\|_1} \right\} \tag{1}$$

We want to maximize α to obtain the largest decline of total weight $\left\|\mathbf{w}^{(t)}\right\|$ when tag q is asked.

We now define the expected value of each tag q at the t-th iteration as

$$\mathbb{E}(q) = \left\|\mathbf{w}^{(t)}\right\|_1^{-1} \langle \mathbf{w}^{(t)}, \mathbf{X}_q \rangle \tag{2}$$

From the property

$$\frac{1}{\|\mathbf{w}^{(t)}\|_1} (\langle \mathbf{w}^{(t)}, \mathbf{X}_q \rangle + \langle \mathbf{w}^{(t)}, 1 - \mathbf{X}_q \rangle) = 1 \tag{3}$$

we know that searching for q that maximizes α is actually maximizing $\min(\mathbb{E}(q), 1 - \mathbb{E}(q))$, which is also equivalent to minimizing $|\mathbb{E}(q) - 0.5|$. Thus the greedy approach is in fact searching for q such that

$$q = \arg\min_q |\mathbb{E}(q) - 0.5| \tag{4}$$

C. Interactive Heuristic Search

Next we propose a heuristic algorithm which is basically an extension of the greedy algorithm.

Suppose at the beginning of each round we consider k future questions in advance. Similar to the greedy algorithm, our goal is to reduce $\|W\|_1$ as much as possible. The difference is that we only consider the worst case to construct the heuristic function.

Let the k question tags be $Q_k = \{q_1, q_2, \ldots, q_k\}$ and the corresponding k answers be $\mathbf{l} = l_1 l_2 \ldots l_k \in \{0,1\}^k$, where 1 stands for "yes" and 0 stands for "no".

Assume that after answering the t-th question, the weight of the i-th item is $w_i^{(t)}$. Then at the end of the $(t+1)$-th round, we have

$$w_i^{(t+1)} = \begin{cases} w_i^{(t)} \gamma^{1-x_{iq_t}} & \text{if the answer is "yes"} \\ w_i^{(t)} \gamma^{x_{iq_t}} & \text{if the answer is "no"} \end{cases} \quad (5)$$

where q_t is the tag contained in the t-th question.

To simplify, for answer $l \in \{0,1\}$, we have

$$w_i^{(t+1)} = w_i^{(t)} \gamma^{l \oplus x_{iq_t}} \quad (6)$$

Therefore, after answering all k questions, the weight of the i-th item is:

$$w_i \prod_{j=1}^{k} \gamma^{l_j \oplus x_{ij}} \quad (7)$$

Hence the total decline of the total weight according to the answer vector $\mathbf{l}$ is

$$W[Q|\mathbf{l}] = \sum_i w_i (1 - \prod_{j=1}^{k} \gamma^{l_j \oplus x_{ij}}) \quad (8)$$

Since $\mathbf{l} \in \{0,1\}^k$, there are 2^k possible answer sequences. To construct the heuristic function, we consider the worst case that produces minimal reduce of weight. Thus the questions should be selected as:

$$Q^* = \arg\max_{Q} \min_{\mathbf{l} \in \{0,1\}^k} \{\sum_i w_i (1 - \prod_{j=1}^{k} \gamma^{l_j \oplus x_{ij}})\} \quad (9)$$

We then employ a heuristic search to find Q^*. Since the algorithm askes for user's response in every round, we call it as *Interactive Heuristic Search*.

V. Theoretical Analysis

In this section we present some degenerated cases for the heuristic search discussed above. We first discuss the case when $k = 2$, that is, 2 future questions in advance in every round. We then present an even simpler case when $k = 2$ and $p = 0$, that is, we assume that the answer always reflects the truth. We let the discount factor $\gamma = 0$, under this condition the heuristic function will degenerate to set functions.

At the end of this section we prove a bound on the ranking of target items given the questions and corresponding answers.

The Behavior of the Algorithm when k = 2. Next we consider a simpler case of heuristic search, namely we search for $k = 2$ tag questions in one iteration. We make the deduction in a different way from the previous one, which results in a simpler heuristic function to be calculated.

Let q_1 and q_2 be the first two selected tags. l_1 and l_2 are the user's corresponding answers.

According to user's answer to q_1, we evaluate the decline of $\|\mathbf{w}\|_1$ as

$$W[q_1|l_1 = 1] = (1-\gamma)\langle \mathbf{w}, \mathbf{1} - \mathbf{X}_{q_1} \rangle \quad (10)$$
$$W[q_1|l_1 = 0] = (1-\gamma)\langle \mathbf{w}, \mathbf{X}_{q_1} \rangle \quad (11)$$

where we use $W[q|l]$ to denote the decline of the weight caused by tag q given the user's answer l.

For simplicity of the deduction and equations, we construct another weight vector $\mathbf{w}'$ and define $w'_i = w_i \gamma^{x_{iq_1}}$. We also let $W = \|\mathbf{w}\|_1$, $W' = \|\mathbf{w}'\|_1$, $W_0 = (1+\gamma)W$, and $W_a = \langle \mathbf{w}, \mathbf{X}_a \rangle$ for a tag a. Another important notation is $W_{a \cap b} = \sum_i w_i (x_{ia} \wedge x_{ib})$. Thus we can simplify Eqs. 10 - 11 as

$$W[q_1|l_1 = 1] = (1-\gamma)(W - W_{q_1}) \quad (12)$$
$$W[q_1|l_1 = 0] = (1-\gamma)W_{q_1} \quad (13)$$

Similarly we evaluate the total decline of weight after asking q_1 and q_2 as

$$W[q_1 \,\&\, q_2|l_1 = 1,\ l_2 = 1] = (1-\gamma)(W_0 - \gamma W_{q_1} - \gamma W_{q_2} - (1-\gamma)W_{q_1 \cap q_2}) \quad (14)$$
$$W[q_1 \,\&\, q_2|l_1 = 0,\ l_2 = 1] = (1-\gamma)(W + \gamma W_{q_1} - W_{q_2} + (1-\gamma)W_{q_1 \cap q_2}) \quad (15)$$
$$W[q_1 \,\&\, q_2|l_1 = 1,\ l_2 = 0] = (1-\gamma)(W + \gamma W_{q_2} - W_{q_1} + (1-\gamma)W_{p_1 \cap q_2}) \quad (16)$$
$$W[q_1 \,\&\, q_2|l_1 = 0,\ l_2 = 0] = (1-\gamma)(W_{q_1} + W_{q_2} - (1-\gamma)W_{q_1 \cap q_2}) \quad (17)$$

Hence our selected question tags q_1 and q_2 is:

$$q_1, q_2 = \arg\max_{q_1, q_2} \min\{Eqs.14 - 17\} \quad (18)$$

which is a clearer and cleaner heuristic function.

The Behavior of the Algorithm when p = 0. Next we consider an even simpler case when $k = 2$ and $p = 0$. Also we let $\gamma = 0$. The intuition for this condition is that, the answerer never lies, and we exclude the items immediately if they contain some tag that the target item doesn't contain or vice-versa. We will see that in this case, the heuristic function degenerates into a set of set functions. We simply replace γ with 0 in the previous heuristic function, then we have

$$W[q_1 \,\&\, q_2 | l_1 = 1,\ l_2 = 1] = W - W_{q_1 \cap q_2} \quad (19)$$
$$W[q_1 \,\&\, q_2 | l_1 = 0,\ l_2 = 1] = W - W_{q_2} + W_{q_1 \cap q_2} \quad (20)$$
$$W[q_1 \,\&\, q_2 | l_1 = 1,\ l_2 = 0] = W - W_{q_1} + W_{q_1 \cap q_2} \quad (21)$$
$$W[q_1 \,\&\, q_2 | l_1 = 0,\ l_2 = 0] = W_{q_1} + W_{q_2} - W_{q_1 \cap q_2} \quad (22)$$

Actually the weights of all items are either 0 or 1, i.e., the weights become binary. We let S be the set of items that currently have weight 1, let P_1 and P_2 be 2 sets covering the items that have the tag q_1 and q_2 respectively, then W becomes the Cardinality function, i.e., we have

$$W = Card(S) \quad (23)$$
$$W_{q_1} = Card(P_1) \quad (24)$$
$$W_{q_2} = Card(P_2) \quad (25)$$
$$W_{q_1 \cap q_2} = Card(P_1 \cap P_2) \quad (26)$$

And declined weight becomes

$$W[q_1 \,\&\, q_2 | l_1 = 1,\ l_2 = 1] = Card(S) - Card(P_1 \cap P_2) \quad (27)$$
$$W[q_1 \,\&\, q_2 | l_1 = 0,\ l_2 = 1] = Card(S) - Card(P_2) + Card(P_1 \cap P_2) \quad (28)$$
$$W[q_1 \,\&\, q_2 | l_1 = 1,\ l_2 = 0] = Card(S) - Card(P_1) + Card(P_1 \cap P_2) \quad (29)$$
$$W[q_1 \,\&\, q_2 | l_1 = 0,\ l_2 = 0] = Card(P_1) + Card(P_2) - Card(P_1 \cap P_2) \quad (30)$$

and it follows that

$$q_1, q_2 = \arg\min_{q_1, q_2} \max \left\{ \begin{array}{c} Card(P_1 \cap P_2), \\ Card(P_1 - P_2), \\ Card(P_2 - P_1), \\ Card(S - P_1 \cup P_2) \end{array} \right\} \quad (31)$$

which is quite intuitive: two sets P_1 and P_2 separate the hypothesis space into several parts, we then search for the sets such that the maximum of these parts is minimized.

The Ranking of the Target. We assume that the number of questions we ask is t. Moreover, we let α be the lower bound of the ratio of the decline over all iterations. More specifically, we let $W^{(1)}, W^{(2)}, \ldots, W^{(t)}$ be the the total weights after each iteration, $l_1, l_2, \ldots, l_t \in \{0, 1\}$ be an arbitrary answer sequence. Then we have

$$\alpha = \inf_{j, l_1, l_2, \ldots} \frac{\sum_i w_i^{(j)} (1 - \gamma^{l_{j+1} \oplus x_i})}{W^{(j)}} \quad (32)$$

Theorem 1 : Let p be the probability that the user answers incorrectly. Let γ be the discount factor. Also we use a sequence of random variables y_i to denote whether the user answers incorrectly in the i-th iteration. If the user answers incorrectly in the i-th iteration, $y_i = 1$, otherwise $y_i = 0$. Namely

$$\Pr[y_i = 1] = p \quad (33)$$

Let $Y_t = \sum_{i=1}^{t} y_i$ and $y = \ln(\gamma) Y_t - t \ln(1-\alpha)$, the target item will be surely in top L items with highest weights where

$$L = \frac{n}{e^y} \quad (34)$$

the expected value of which is:

$$\mathbb{E}[L] = n[(1-\alpha)(1 + \frac{p}{\gamma}(1-\gamma))]^t \quad (35)$$

which means, the target item will be surely in top $L = \frac{n}{e^y}$ items with highest weights where L is a random variable, and the expected value of L is $n[(1-\alpha)(1 + \frac{p}{\gamma}(1-\gamma))]^t$.

However, this bound depends on α, which is hard to evaluate when we don't know the weight sequence $W^{(1)}, \ldots, W^{(t)}$. Only when we get whole sequence of weights are we able to get the value of α, and subsequently get the bound for the expected value of L.

Next we will show a bound on the expected value of L on a random tag matrix, which doesn't depend on α.

Suppose we know the probability of the entry in the tag matrix being 1 is q. Namely the tag matrix is generated with each entry being 1 with probability q and 0 with probability $1-q$. Moreover, the value of each entry is set independently.

Then we have the following theorem:

Theorem 2 : After answering t questions, the target item will be surely in top L items with highest weights, and the expected value of L has the following bound:

$$\mathbb{E}[L] \leq \frac{(2r)^t n}{2} e^s + 1 \quad (36)$$

where

$$r = ((2-4p)q^2 - (2-4p)q + 1 - p) \quad (37)$$
$$s = -t/4 + p(t-1)(1-p) \quad (38)$$

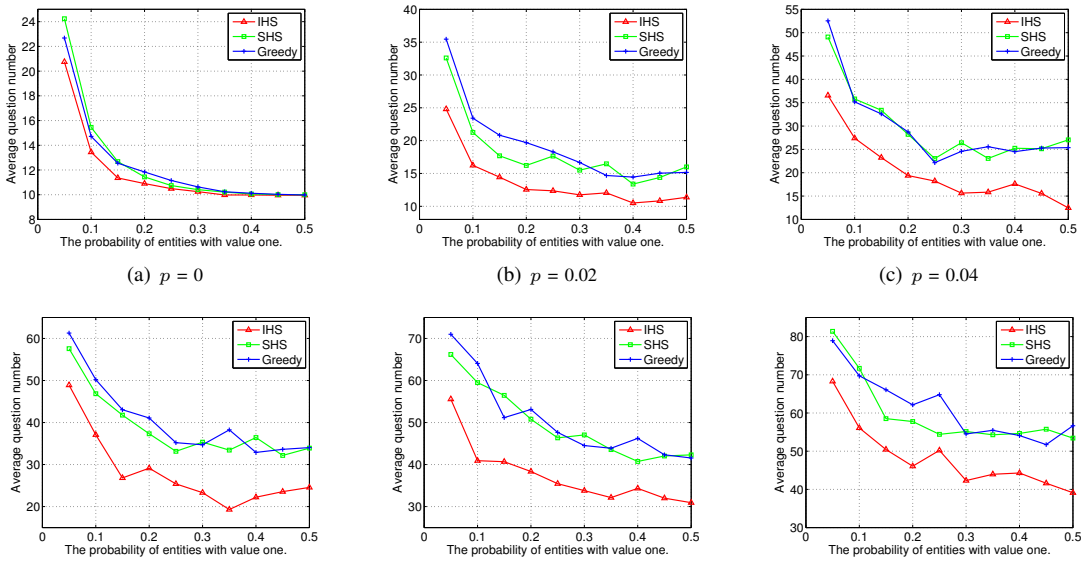
We can see that this bound on the expected value of L doesn't depend on α, thus easier to evaluate.

The proof of Theorem 1-2 can be found in the appendix.

VI. Experiments

A. Experimental Setup

We perform experiments on 3 data sets to compare the performance of Interactive Heuristic Search (IHS), Greedy, and Static Heuristic Search (SHS), the details of the former two algorithms can be found in Section IV. To show the power of interaction with the user, we propose the third algorithm SHS. The question selection method of this algorithm is the same as IHS, but it only interacts with the



(a) p = 0 (b) p = 0.02 (c) p = 0.04

(d) p = 0.06 (e) p = 0.08 (f) p = 0.1

Figure 2. The performance of different algorithms in Random data set.

user every 2 rounds, in one iteration we search for $k = 2$ question tags, use both of them as question tags and present them sequentially to the answerer. Note that during these 2 iterations, the system doesn't have any interaction with the user, i.e., the answer of the user in the first iteration doesn't affect how the system choose the question tag in the second iteration.

Random. In this data set, the tag matrix is randomly constructed, with 1000 items, 100 tags, and each entry is initialized to 1 with a manually defined probability q.

Patent. In this data set, we extract 13,091 companies and 3,770,411 patents from USPTO[1]. Companies are regarded as items and 428 categories of patents are tags. We then conduct the tag matrix as follows: if one company owns at least one patent belongs to a specific category, the corresponding entity is 1, otherwise it is 0. Each row in the tag matrix contains 17.2 entities with value 1 on average.

ArnetMiner. This data set consists of 36,371 conferences in computer science and 1,905,496 papers provided by ArnetMiner[2]. We then generate distributions θ_v for each conference v on 200 topics by ACT model [24]. We treat each conference as an item and each topic as a tag. We set the entity in tag matrix $x_{vz} = 1$ if $\theta_{vz} > \beta$, where z is a topic and β is a threshold manually defined. In the tag matrix, each row contains 21.2 entities with value one on average.

For each data set, we build a simulator of the answerer. At first, the answerer select an item as the target. Then in each round, the answerer is given a question q generated by the question selection algorithm (questioner). The answerer responses the answer to the questioner according to the value of x_{qv}. The interactive progress continues until some stop condition is met, e.g., the target ranks first, in other words, the weight of the target item is the largest and no other items have the same weight as the target item.

[1]http://www.uspto.gov/

[2]http://arnetminer.org/

B. Quantitative Analysis

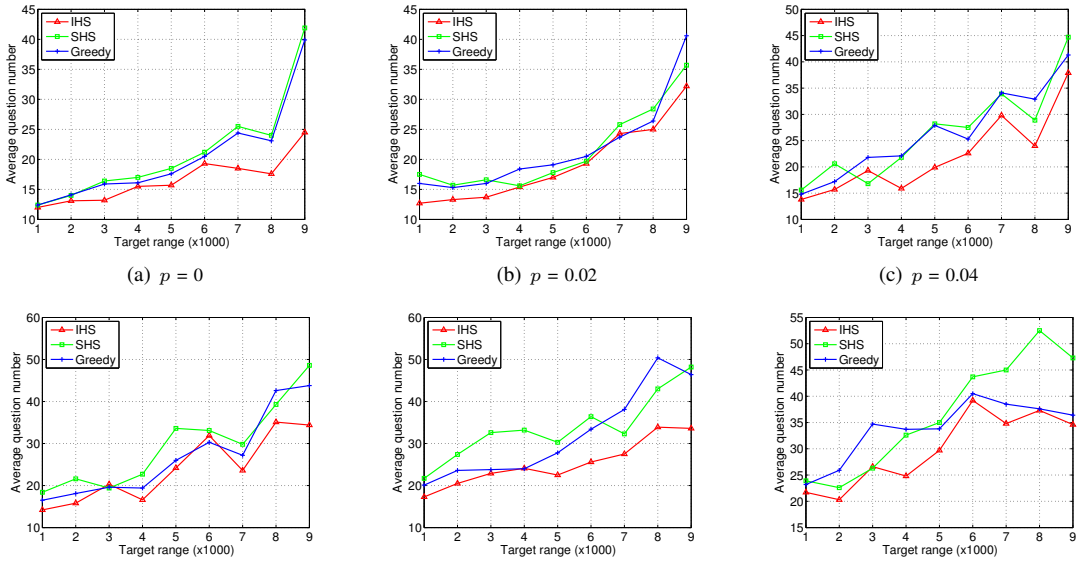
In the Random data set, we randomly pick up 300 items as the target one by one and count the average number of questions used to identify a target, denoted as r. We vary the probability q from 0.05 to 0.5, and see how performance of different algorithms change. In the Patent data set, we first count the number of tags which "cover" the i-th item as

$$C(i) = \sum_{j=1}^{m} X_{ij} \tag{39}$$

We then rank all items according to $C(i)$. After that, we first select top 1000 items as the targets, then select top 2000 items and keep continue. Similar experiments are conducted on ArnetMiner data set. Besides, we also vary the p and see how user's mis-operations affect the results. Figures 2 - 4 show the detailed results on 3 data sets respectively.

Firstly, we can see that IHS outperforms Greedy and SHS (8.07 and 8.86 less questions on average respectively), especially when the user answers incorrectly with higher probability, e.g., IHS uses 12.7 less questions than Greedy in Figure 3(f). In most cases, SHS requires more questions than Greedy (31.85 Vs. 31.06 on average), which indicates that it is important to keep interacting with the user.

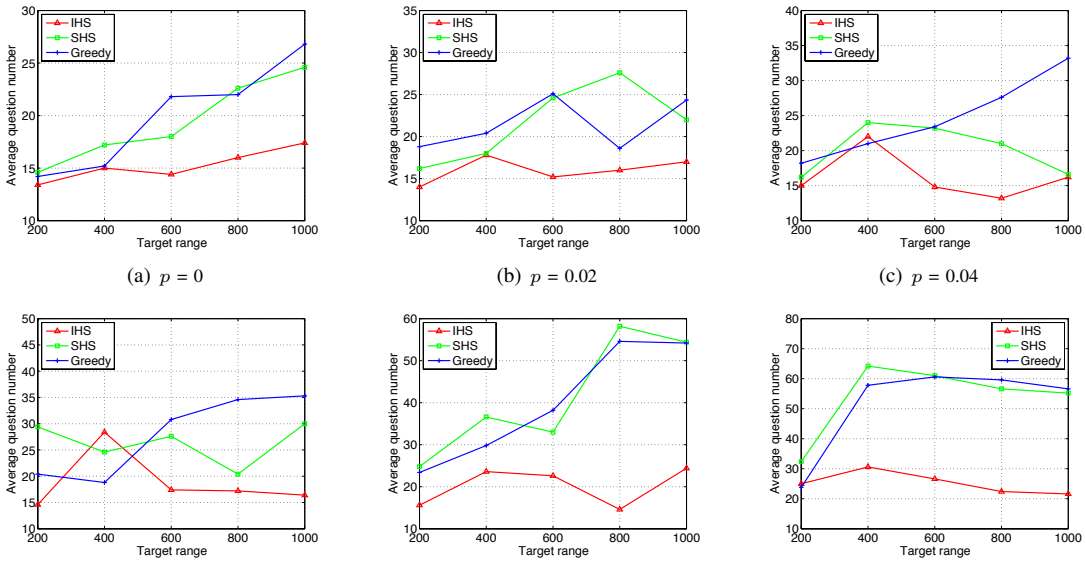
Secondly, from Figure 2, we can see that basically r decreases as q increases. The reason is quite intuitive: if



(a) $p = 0$ (b) $p = 0.02$ (c) $p = 0.04$

(d) $p = 0.06$ (e) $p = 0.08$ (f) $p = 0.1$

Figure 3. The performance of different algorithms in Patent data set.



(a) $p = 0$ (b) $p = 0.02$ (c) $p = 0.04$

(d) $p = 0.06$ (e) $p = 0.08$ (f) $p = 0.1$

Figure 4. The performance of different algorithms in ArnetMiner data set.

the tag matrix is very sparse, i.e., each item has very few tags, it is hard for the algorithm to find the tag that the item contains, thus it costs more questions to find the target item. In Figure 3 and Figure 4, r also decreases as the target range expand, which can be explained in the same way.

Thirdly, from all figures, we can easily find that r increases as p increases. This indicates that, if the user knows not that much about target item and keeps answering incorrectly, it is hard for the questioner to find the answer. We can also see curves in all figures tend to be monotonous and more smooth with smaller p.

Another interesting fact is that, the gap between the performance of IHS and Greedy becomes larger as p increases. By a careful investigation, we find that when user answers incorrectly, the weights of items which match the answer and ranks higher than the target item do not change. We denote these items by set S and the target item is not in S. To recover from this fault, the questioner must ask questions which can differ the target item from items in S, otherwise the target will never rank higher than items in S.

Table I
EFFICIENCY STATISTIC (IN SECONDS).

Data set	IHS	SHS	Greedy
Random	0.025	0.009	0.006
Patent	0.321	0.299	0.087
ArnetMiner	0.120	0.149	0.115

Figure 6. A screenshot of the prototype system.

IHS performs better than Greedy since IHS considers more tags in one iteration, which makes it more probable to select the tag can differ the target and items in S. It also indicates that IHS has a stronger capability to anti mis-operations and has a more robust performance than Greedy.

C. Efficiency Analysis

To see how efficient our approach is, we count the average time used by Interactive Heuristic Search, Static Heuristic Search, and Greedy in each interaction round. Table I shows the results. In all data sets, IHS requires more time than Greedy and Static Heuristic Search. However it takes less than 1 second (0.16 second in average) which can be tolerated in real applications. Static Heuristic Search costs almost half time of Interactive Heuristic Search as it generates a new pair of questions in every two rounds.

D. Qualitative Analysis

Now we present a case study to demonstrate the effectiveness of the proposed approach. Figure 5 shows an example generated from our experiments. It represents a portion of interactive rounds when the target item is IBM. Several parameters and tags used in questions generated by different algorithms are showed in the figure, where $|W|$ denotes the sum of weights before the corresponding round starts, *rank* stands for the rank of the target item, and E denotes the expectated value of the selected tag. Black arrows indicate the user's responses.

In the first round, Greedy selects a tag with expectation value close to 0.5 which is reasonable. But in the next round, it underperforms IHS, since all the tags cannot partition the hypothesis space well. IHS makes a relatively worse choice at first but turn to a favorable situation in next. SHS chooses the same tag with IHS at first but it does not interacts with the user after this round, thus loses helpful information. After the first two rounds, IHS reduces the weights most and makes the target ranks top 2.

E. Prototype System

We have developed and deployed a web application for interactive user intention understanding based on our approach and Patent data set in PatentMiner[3] []. Figure 6 shows a screenshot of the prototype system. Users can find suitable companies for job hunting or business analysis. From experimental results, we can see that in Patent data set, IHS requires around 14 questions to identify the target. But in real applications, it uses less questions as the system displays top 5 companies during each round, thus the user can find the target even when it does not rank the first. Throughout the experiments, we find that the questioner averagely requires 2-3 questions to make an item ranks first from top 5. Also, the system allows users to enter a query and search for some candidates first to limit the size of hypothesis space and reduce the number of questions required. At present, we've started collecting users' feedbacks from the prototype system to further improve our work.

VII. CONCLUSION AND FUTURE WORK

In this paper, we study a new problem, Interactive User Intention Understanding, to help users identify their target items. A heuristic search based algorithm is designed to generate questions effectively. We also give a bound on the ranking of the target item after the user has been asked a number of questions. At last we conduct a series of synthetic experiments on three data sets and show that our approach outperforms two baseline methods.

This is our first attempt in interactive user intention understand, which remains quite a few future work. One challenge of this work is how to reduce time cost in each interaction round, which limits total time of searching in our algorithm. An interesting and challenging idea is: can we build up a decision tree corresponds to questions we will use according to user's response? In this way it only cost $O(1)$ time to choose a question in each round, which sounds quite exciting.

Another idea is that, in the real world, more popular an item is, more likely it may be the target. Thus we can first make an assumption of the distribution over all possible

[3]http://pminer.org

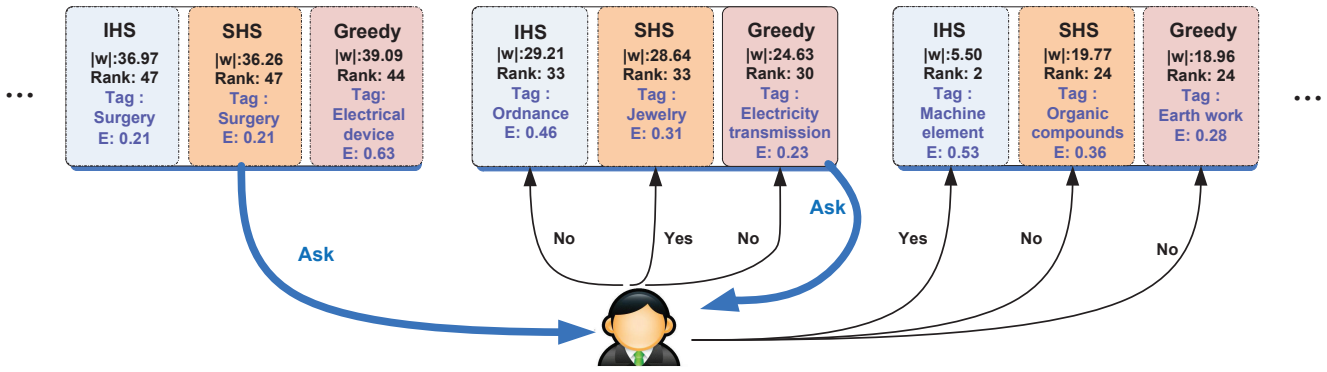


Figure 5. Case study. Portion of interactive rounds when the target company is IBM. $|W|$ denotes the sum of weights before the corresponding round starts, *rank* stands for the rank of the target item, *tag* is the selected tag used to generate questions, and E denotes the expectation value of the selected tag. Black arrows indicate the user's responses.

target items, and use the knowledge of this distribution to ask smarter questions.

Acknowledgements

The work is supported by the National High-tech R&D Program (No. 2014AA015103), National Basic Research Program of China (No. 2014CB340506, No. 2012CB316006), Natural Science Foundation of China (No. 61222212), National Social Science Foundation of China (No. 13&ZD190), NSF CAREER Award (No. 1453800), NSFC-ANR (No. 61261130588), and a research fund supported by Huawei Inc.

We thank Chengtao Li and Xun Zhen for valuable discussions and suggestions. Chengtao also contributed to the theoretical analysis of the proposed algorithm.

References

[1] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, A. Verkamo, et al. Fast discovery of association rules. *Advances in knowledge discovery and data mining*, 12:307–328, 1996.

[2] M. Bauer, D. Dengler, and G. Paul. Instructible information agents for web mining. In *Proceedings of the 5th international conference on Intelligent user interfaces*, pages 21–28. ACM, 2000.

[3] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006.

[4] Z. Chen, F. Lin, H. Liu, Y. Liu, W. Ma, and L. Wenyin. User intention modeling in web applications using data mining. *World Wide Web*, 5(3):181–191, 2002.

[5] T. Denoeux. A k-nearest neighbor classification rule based on dempster-shafer theory. *Classic works of the Dempster-Shafer theory of belief functions*, pages 737–760, 2008.

[6] S. Dudani. The distance-weighted k-nearest-neighbor rule. *Systems, Man and Cybernetics, IEEE Transactions on*, (4):325–327, 1976.

[7] D. Fragoudis and S. Likothanassis. Retriever: An agent for intelligent information recovery. In *Proceedings of the 20th international conference on Information Systems*, pages 422–427. Association for Information Systems, 1999.

[8] R. Graham, D. Knuth, and O. Patashnik. *Concrete mathematics: a foundation for computer science*, volume 2. Addison-Wesley Reading, MA, 1994.

[9] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse. The lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 256–265. Morgan Kaufmann Publishers Inc., 1998.

[10] W. Hunt, W. Mark, and G. Stoll. Fast kd-tree construction with an adaptive error-bounded heuristic. In *Interactive Ray Tracing 2006, IEEE Symposium on*, pages 81–88. IEEE, 2006.

[11] B. J. Jansen, A. Spink, and J. O. Pedersen. A temporal comparison of altavista web searching. *JASIST*, pages 559–570, 2005.

[12] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, Sept. 1999.

[13] D. Lemire and A. Maclachlan. Slope one predictors for online rating-based collaborative filtering. In *Proceedings of SIAM Data Mining (SDM'05)*, 2005.

[14] R. Lempel and S. Moran. Salsa: the stochastic approach for link-structure analysis. *ACM Trans. Inf. Syst.*, 19(2):131–160, Apr. 2001.

[15] L. Lovász, J. Pelikán, and K. Vsztergombi. *Discrete mathematics: elementary and beyond*. Springer Verlag, 2003.

[16] A. Moore. An intoductory tutorial on kd-trees. *Extract from Andrew Moore's PhD Thesis: Effcient Memory based Learning for Robot Control*, 1991.

[17] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999.

[18] G. Pass, A. Chowdhury, and C. Torgeson. A picture of search. In *Infoscale'06*, 2006.

[19] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, WWW '01, pages 285–295. ACM, 2001.

[20] B. Settles. Active learning literature survey. *Computer Sciences Technical Report 1648*, 2009.

[21] C. Silverstein, M. R. Henzinger, H. Marais, and M. Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, pages 6–12, 1999.

[22] R. Slavin. Cooperative learning. *Review of Educational research*, 50(2):315–342, 1980.

[23] J. Spencer. Balancing games. *J. Combin. Theory Ser. B*, 1977.
[24] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su. Arnetminer: extraction and mining of academic social networks. In *KDD'08*, pages 990–998, 2008.
[25] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2:45–66, 2002.

VIII. Appendix

A. Theorem 1

Proof: Since in t iterations the user answers Y_t questions incorrectly, the weight of the target item is γ^{Y_t}. We know that at the t-th iteration, the total weight $W^{(t)}$ is no greater than $W^{(0)}(1-\alpha)^t = n[1-\alpha]^t$, because at least $\alpha W^{(i-1)}$ vanishes after answering the i-th question.

The target item will be surely ranking in top L after the t-th iteration if

$$L \times \gamma^{Y_t} \geq W^{(t)} = n(1-\alpha)^t \tag{40}$$

We could simply take

$$L = \frac{n(1-\alpha)^t}{\gamma^{Y_t}} \tag{41}$$

To get the expected value of L, we have

$$\begin{aligned}
\mathbb{E}[L] &= \mathbb{E}[n(1-\alpha)^t \gamma^{-Y_t}] && (42)\\
&= n(1-\alpha)^t \mathbb{E}[(\frac{1}{\gamma})^{Y_t}] && (43)\\
&= n(1-\alpha)^t \prod_{i=1}^{t} \mathbb{E}[\frac{1}{\gamma}^{y_i}] && (44)\\
&= n(1-\alpha)^t [1 + \frac{p}{\gamma}(1-\gamma)]^t && (45)
\end{aligned}$$

which is exactly the result we are looking for. ■

B. Theorem 2

Proof: We let the user answers $l_1, l_2, \ldots, l_t$ to these t questions where $l_i \in \{0, 1\}$. Assume that the target item is v (the v-th item). Using the definition of p, we have

$$\begin{aligned}
\Pr[l_i = x_{vi}] &= 1-p && (46)\\
\Pr[l_i = 1 - x_{vi}] &= p && (47)
\end{aligned}$$

Since each entry is set to be 0 or 1 independently, for $j \neq v$,

$$\begin{aligned}
\Pr[l_i = x_{ji}] &= p \times 2q(1-q) + \\
&\quad (1-p) \times (q^2 + (1-q)^2) = r && (48)\\
\Pr[l_i \neq x_{ji}] &= p \times (q^2 + (1-q)^2) + \\
&\quad (1-p) \times 2q(1-q) = 1-r && (49)
\end{aligned}$$

and it follows that $r > 1 - r$

We let N_j be the number of tags that in which item j is different from $l_1, l_2, \ldots, l_t$. Then we have

$$\begin{aligned}
\Pr[N_j = w] &= \binom{t}{w}(1-r)^w r^{t-w} && (50)\\
&< \binom{t}{w} r^t && (51)
\end{aligned}$$

An item $j \neq v$ ranks higher than the target item v if and only if $N_j < Y_t$. The intuition for this is that item j has more tags that match the user's answers $l_1, \ldots, l_t$ than item v. Therefore we have

$$\begin{aligned}
\Pr[N_j < Y_t] &= \sum_{i=0}^{Y_t - 1} (1-r)^i r^{t-i} \binom{t}{i} && (52)\\
&< r^t \sum_{i=0}^{Y_t - 1} \binom{t}{i} && (53)\\
&< r^t \times 2^{t-1} \frac{\binom{t}{Y_t}}{\binom{t}{\frac{t}{2}}} && (54)\\
&\leq \frac{(2r)^t}{2} e^{-\frac{(t/2 - Y_t)^2}{t}} && (55)
\end{aligned}$$

Eqs. 54-55 draw from [15] [8].

Since the rank of the target item L is exactly the number of items that rank higher than it plus 1, we have

$$\begin{aligned}
\mathbb{E}[L] &= n \times \mathbb{E}[\Pr[N_j < Y_t]] + 1 && (56)\\
&\leq \frac{(2r)^t n}{2} \mathbb{E}[e^{-\frac{(t/2 - Y_t)^2}{t}}] + 1 && (57)\\
&\leq \frac{(2r)^t n}{2} e^{\mathbb{E}[\frac{-(t/2 - Y_t)^2}{t}]} + 1 && (58)\\
&= \frac{(2r)^t n}{2} e^{-t/4 + p(t-1)(1-p)} + 1 && (59)\\
&= \frac{(2r)^t n}{2} e^s + 1 && (60)
\end{aligned}$$

where Eq. 58 draws from Jensen's Inequality.

■