A Cognitive User Model for E-Commerce Search

Sahiti Labhishetty* Chengxiang Zhai* Suhas Ranganath† Pradeep Ranganathan‡

Abstract

We present a novel cognitive user model for E-Commerce (E-Comm) 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 model all the major search behavior of a user, including query formulation, clicking on results, and query reformulation, and thus provides a complete model for users of E-Comm search. The model has interpretable parameters that can be estimated using E-Comm search log data to analyze and understand users' behavior as well as can be manually adjusted to simulate different kinds of users. The trained user model using the search log can also be used to simulate users or analysis of search log. Preliminary evaluation on an E-Comm search log shows that the model performs well for predicting user behavior in a search session. The analysis of search sessions reveals multiple interesting findings about the search behavior of E-Comm search users.

1 Introduction

A formal User Model has multiple applications, it can be used for user simulation which can be used for evaluating interaction search system, to analyse user behaviours search sessions, and to optimize search engine interaction with the user. In this paper, we propose a conditional generative user model for mining E-Comm search log to understand users' behavior, with an emphasis on modeling user's cognition by modelling user's changing knowledge, information need, and query reformulation. The main idea of the proposed model is the generation of all important user actions in a whole user session 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: 1)it explicitly models the user cognitive state and update of the state to model different user behaviour characteristics in one unified model. 2)the query formulation/reformulation model is studied in depth and the query reformulations in search log are also used for analysis. 3)The interpretable model parameters can be learned from the search log and it can be fit to even single session to identify specific user behaviour. Our model has several benefits, it can be used (a) as a tool to identify interesting user behaviour patterns in search log (b) to generate or simulate user sessions given an input information with varying user behaviours by altering the parameters. We evaluate the model on an E-comm search log and also show that the model can be used to analyze user behavior in multiple ways. Our results show that (1) users generally differ in exploration/non-exploration behaviour (2) the variance in behaviour is larger for same user purchasing different products than compared to that of different users buying same product. The model can have many other uses including user simulation which can in turn be be used to evaluate interactive search systems, it can be used to optimize the search engine interaction with a user.

In E-commerce, there are only only few works [4] analysing queries and user behavioural patterns through search log. In web search, there are works analysing or predicting user purchase or browsing behaviour [5, 6, 7, 8] but these works do not generalize to E-comm search log. The user simulation models in Web search[3, 2, 1] cannot be generalized to generate user interaction in E-comm. None of the above works consider cognitive aspect of the user in E-comm and lack the novel factors of our work described above.

2 A Cognitive State User Model (CSUM)

The proposed formal cognitive state user model (CSUM) is meant to formally model how an E-Comm user searches. Since a user generally starts with an information need (IN) in mind, CSUM is a conditional generative model conditioned on some assumed IN and knowledge which 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 which can be potentially repeated multiple times. We now describe CSUM

*University of Illinois, Urbana-Champaign, USA. {sahitil2,czhai}@illinois.edu

†WalmartLabs, India. Suhas.Ranganath@walmartlabs.com

‡Lyft,USA. mail.pradeepr@gmail.com

in more detail. Different components of the proposed

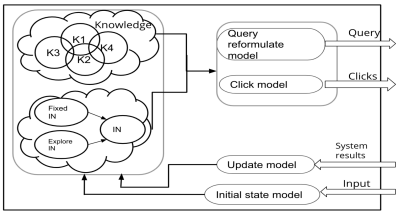

Figure 1: Outline of the proposed CSUM model

model CSUM and how they are connected to interact with search engine are shown in Figure 1.

2.1 Representation of the IN We first need to model the IN in a user's mind that has triggered the search task. In general, we can represent the IN as a set of probability distributions of all those attribute values of products preferred by a user. We consider three basic attributes of a product which most users likely "care about": category (C), brand (B), and title words (Ti), with attribute values $\{c_i\}$,$\{b_i\}$,$\{t_i\}$, respectively. For each attribute value , a Bernoulli (binary) variable (e.g., $V_{C,c_1} \in \{0,1\}$) is used to capture to what extent a user likes or dislikes the attribute value with $p(V_{C,c_1} = 1)$ indicating that probability that the user likes products in the c_1 category.

When we estimate CSUM on a search log, we have access to the final target product purchased by a user, which we can use to estimate the initial IN probability distributions of preferred attribute values by the user. One simple way to define IN is to fix it to a target product (i.e., setting all the attribute value probabilities to 1.0), which we will refer to as a ***Fixed IN*** model. Let the probability distribution be denoted as $P(V_{A,a_i}|fixed\_IN)$ where A is the attribute which can be C, B, Ti and a_i are the attribute values of A. However, *Fixed IN* is only appropriate (accurate) if the user indeed had this particular product in mind and had never changed mind during the search process but there is possibility for uncertainty in preferences. To model this uncertainty, we further introduce an ***Exploring IN*** model where the probability of a target attribute value is still relatively large, but not 1.0, thus leaving room for exploration. The probably distribution for *Exploring IN* is $P(V_{A,a_i}|explore\_IN)$.

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 and denoted by $P(fixed\_IN_A)$ or $P(f_A)$ for short, for attribute A which can be C, B, Ti. The uncertainty is captured by a mixture model with $P(f_A)$ indicating the probability of user being in the state of *Fixed IN*, thus 1-$P(f_A)$ is the probability of being in state of *Exploring IN*. In the extreme case of $P(f_A) = 1$, it would indicate that the user has "made up mind" to go for the target product (with no desire to explore other attribute values).

2.2 Representation of Knowledge State From cognitive IR perspective, the user has knowledge acquired from different types of sources, including knowledge of the product space, knowledge of the search environment, knowledge of how the search system understands his/her query, and general knowledge of words. Accurate modeling of users requires modeling such knowledge; modeling this knowledge is the main novelty of our model as compared with the previous work.

Formally, we represent the knowledge of how a system understands the user query (part of knowledge of the product space) with two conditional distributions: 1) $P(cat/w_q)$ denotes the knowledge about the likelihood that a query word w_q would lead a search engine to return a particular category of products *cat* as retrieval results. 2) $P(w_p/w_q)$ denotes the knowledge about the likelihood that a query word w_q would lead a search engine to return a product with word w_p occurring in the brand field or the title field. Further, a user may have prior knowledge of these mappings based on previous experience or general knowledge of E-commerce product space, we refer to this as ***Background knowledge***(K_1) and it is denoted by $P_{kb}(cat/w_q)$, $P_{kb}(w_p/w_q)$. Alternately the user can learn about this knowledge during the search session using the search results obtained for a query, we refer to this as ***Knowledge Learnt*** and it is denoted by $P_{kl}(cat/w_q)$, $P_{kl}(w_p/w_q)$. The *Knowledge Learnt*(K_2) is a dynamic variable and is updated throughout the session as the user interacts more with the search engine and it is specific to the information need of the user as the queries are specific to the IN.

Further, the knowledge of the product space also involves knowing the text description of the attribute values as they are described in the product space. The user may also know that the search system also uses keyword matching to obtain the results. Therefore words that are present in the attribute values is considered as ***Keyword matching knowledge***(K_3). This can be represented either as a set of words of all attribute values that are in the context of IN: words in $\{V_{A,a_i}\}$ or as a probability distribution $P_{key\_m}(w|w_a)$ where implies $P_{key\_m}(w|w_a)$ is 1 if $w = w_a$ and is 0 for other words and w_a belongs to set of words of all attribute values in context of IN.

In addition to knowledge of product space, the user further has ***Linguistic knowledge***(K_4) of word meanings. We model this knowledge by approximating the word similarity of the user to that of the word similari-

ties obtained through word embeddings. The word similarity score is normalized to make a probabilistic model similar to K_1, K_2. For a word w, the similarity between w and all other words in the context of IN are computed and the top most word similarities are selected and normalized to form the distribution $P_{sim}(w_1|w_2)$.

2.3 Update model The user state variables, IN and knowledge are updated in the following way using the *Update model* to simulate what happens to a user during a session.

2.3.1 Information need update The IN can be updated either after clicking or skipping each product in the results page. In the current model, we update the IN 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 query or purchasing or stopping the session.

Formally, if the user observes more attribute values from the *Fixed IN*, they prefer them more and therefore will be closer to *Fixed IN* and this is captured by increasing the value of $P(fixedIN)$. $P(f_A)$ is updated for each attribute A if the attribute values of A are observed. The update is proportional to the probability of preference of those attribute values. For example, for attributes like brand the $P(V_{B,b_i}|fixed\_IN)$ can be high and viewing that brand attribute value b_i may increase the $P(f_B)$ to a greater extent. For some of the title word attributes which are not very crucial, the $P(V_{T,Ti_i}|fixed\_IN)$ can be low, and observing those attributes may not influence the user so much to update the IN, therefore the increase in $P(f_{Ti})$ can be less. Further, the amount of update also depends on the user preference to explore or not to explore, even though the user observes the attribute from *Fixed IN*, the user might choose to explore more before deciding to prefer it. This indicates the user behaviour to explore or not and we capture this using a parameter, let this be λ_1. The increase in $P(f_A)$ for attribute A based on observed attributes $\{o_i\}$ is, $P_o(f_A)$

As the session length increases, the user fixes his/her product preferences and the probability of *Fixed IN* should increase as the end point is target product T. $P(f_A)$ is updated along each attribute A and the amount of update is directly proportional to session length l. Similar to λ_1, there is an additional parameter that influences the amount of update based on session length, let this be λ_2. This parameter also indicates user preference to explore, specifically it indicates if the user wants to explore for more time even with a long session. But unlike λ_1, λ_2 is inversely proportional to increase in the $P(f_A)$. The increase in $P(f_A)$ along each attribute A based on session length l is $P_{session}(f_A)$ and is computed as follows. Note that this update is same for all the attributes. Therefore $P(f_A)$ is updated using $P_o(f_A)$ $P_{session}(f_A)$. These probabilities are computed as

$P_o(f_A) \propto \lambda_1 * P(V_{A,\{o_i\}}|fixed\_IN)$ where $\{o_i\}$ is the set of observed attribute values among the results.

$P_{session}(f_A) = \frac{l}{f(\lambda_2)+l}$ where $f(\lambda_2)$ is linear function of λ_2, l is the length of the session until then.

Finally, when a user clicks on a product item they usually might update their preferences for the attribute values of the item. The user may like or dislike the product. The user will likely decrease their preference for those attribute values that are different from the *Fixed IN* i.e, attribute values which are not in target product T. $P(V_{A,a_i})$ are updated for a_i in clicked products that are not in T. The amount of decrease in the preference is given by the factor $\alpha_{iupdate}$. Let Clk_p be the set of clicked product item attribute values.

$P(V_{A,a_i}) = P(V_{A,a_i}) * (1 - \alpha_{iupdate}) \forall a_i \in Clk_p \setminus T$

2.3.2 Knowledge Update Similar to IN, the user learns and obtains more knowledge of the product space while interacting with the system. The user's knowledge state is also updated each time after interacting with a whole page of results form the system. We only update *Knowledge learnt* (K_2) component of knowledge of the product space, leaving the exploration of more sophisticated update model as future work. The probability $P_{kl}(cat|w_q)$ is directly proportional to the count of number of products from result page that belong to category cat which 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 result page which have both w_p and w_q among any of its attributes values. Formally, let $f_1(r_p, clk_p)$, $f_2(w_p, r_p, clk_p)$ be the scores 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 the hypothesis, the function in inversely proportional to r_p and directly proportional to clk_p. Let w_{pi} be all the words in the attribute values of product p. Therefore,

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

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

$\forall w_q \in query\,words$ where the $\{P_{w_q,cat}\}$ is set of result products which have w_q in w_{pi} and cat as the category. Similarly $\{P_{w_q,w_p}\}$ is set of results products which have

w_q and w_p in w_{pi}. w'_p are all words in product titles and brand attributes of all result products $\{P\}$.

If the session length is more than 1, then for later session interaction 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*. $\alpha_{kupdate}$ is a parameter used to update the *Knowledge learnt* during the session.

$P_{kl}(cat/w_q) = P_{kl}(cat/w_q) + \alpha_{kupdate} * P_{kl,new}(cat/w_q)$
$P_{kl}(w_p/w_q) = P_{kl}(w_p/w_q) + \alpha_{kupdate} * P_{kl,new}(w_p/w_q)$

2.4 Query Reformulation model With updated representation of IN and knowledge states in place, we now describe how to model query reformulation. We assume the following strategy for the user to reformulate the query. The user thinks about removing query words and adding new query words (only adding words in case of formulating first query). This is done by scoring a query as a function of individual word scores. The user tries to maximize the query score. In order to remove or add a word, the user removes or adds a word that increases the query score to the maximum compared to the current query. Finally based on constraints on the on query score maximization, we choose the removed and added words and resultant query is formed.

Sample space of words: Though the Information need of the user can include many attribute values and words of those values, the user may not know all the words and may not use all the known words for making the query. Therefore, a sample space of words S_{qw} is defined with all the words of the target product T, and the top-k common query words used in the sessions where the final target product belong to same category as T. Note that we do not imply here that the user is aware of all words in S_{qw}. The set S_{qw} can be interpreted in two ways, first the user is aware of all words in S_{qw} but chooses the words based on the score obtained through different Knowledge sources he might use. Second, the user is not aware of all the words in S_{qw} but gets reminded of these words based on the different Knowledge sources. Either interpretation might be true here and conclusion about this is out of the scope of this work.

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*, *linguistic Knowledge*, *Knowledge learnt*, *Keyword matching knowledge* and the *Information need (IN)*. Each knowledge source is given a weight indicating the importance of that source for scoring words and thereby in query reformulation. The score for a potential query word can be understood as a function of the following, the score based on Information need(IN), score from different Knowledge sources $\{K_i\}$ and weight of the knowledge sources $\{\alpha_{K_i}\}$. Therefore the general formulation for the score a word is defined as,

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

where K_i is a knowledge source, v_{A,a_i} is a either a attribute value or word in the attribute value for attribute A and IN is the current IN of 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. **Background Knowledge source(K_1):** In this case, the $K\_score(K_1, w, v_{A,a_i})$ is given by $P_{kb}(cat|w_q)$, $P_{kb}(w_p|w_q)$ for category attribute values and brand/title attribute value words respectively. Therefore, the word score based on K_1 is,

$w\_score_C(w) = \alpha_{K_1} * \sum_{cat} P_{kb}(cat|w) * P(V_{C,cat}|IN)$
$w\_score_A(w) = \alpha_{K_1} * \sum_{w_a} P_{kb}(w_a|w) * P(V_{A,a_i}|IN)$

where attribute A can be B or Ti and and w_a belongs to the words describing of a_i

Knowledge learnt source(K_2): Similar to K_1, the conditional probabilities $P_{kl}(cat|w_q)$, $P_{kl}(w_p|w_q)$ are used for K_score for category and brand/title attributes respectively.

$w\_score_C(w) = \alpha_{K_1} * \sum_{cat} P_{kl}(cat|w) * P(V_{C,cat}|IN)$
$w\_score_A(w) = \alpha_{K_1} * \sum_{w_A} P_{kl}(w_a|w) * P(V_{A,a_i}|IN)$

where attribute A can be B or Ti.

Keyword matching knowledge(K_3): The K_score is 1 if the word belongs to the words describing the attribute v_{A,a_i}, the category attribute is also described in words for computing this score.

$K\_score(K_3, w, v_{A,a_i}) = 1\, if\ w \in words\ in\ v_{A,a_i}$
$= 0\, otherwise$
$w\_score_A(w) = \alpha_{K_3} * K\_score(K_3, w, v_{A,a_i}) * P(V_{A,a_i}|IN)$

where attribute A can be C,B or Ti

Linguistic knowledge score(K_4): As described in Section 2.2, the user may also have *Linguistic Knowledge* of word meaning. The $K\_score(K_4, w, v_{A,a_i})$ is computed based on $P_{sim}(w_1|w_2)$ computed by normalizing the similarities of w_2. Here w_1 can be attribute value word from category, brand or title attribute value description. This similarity is computed only for words w_2 that are outside of product space. Therefore the word score is given by,

$w\_score_A(w) = \alpha_{K_4} * \sum_{w_a} P_{sim}(w_a|w) * P(V_{A,a_i}|IN)$ if $w \notin A$ where attribute A can be C,B or Ti
$= 0$ if $w \in A$ where A can C,B or Ti

Query scoring:
Query score is the function of all the word scores ag-

gregated over all attributes. In the current model, we choose the function as average operation for both word scores and attributes. Therefore, query score is average of attribute scores where attribute score is average of word scores.

$Q\_score(Q) = \sum_A \sum_{w_q \in Q} w\_score_A(w_q)$

To solve for the new query during query reformulation, we solve for words which give maximum Q_score compared to Q_score of current query. Each word is scored independently in this way. Therefore final score of word w is,

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

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 case of first query) and the top ranked words are chosen according to a numerical threshold or top-k threshold.

2.5 Click model The user decides to click on the product when it is according to his/her preferences. In this work, we only explore a preliminary probabilistic model for click which is *Document generation model*. For click decisions we only use user IN variable. Based on the IN, a model for relevant and irrelevant product items are made. The odds of the likelihood of the product by relevant model to irrelevant model gives the probability of clicking it. Note that the IN is binary distribution on all the attribute values of a attribute, we combine the binary distributions across multiple attributes and normalize it to single multinomial distribution. Therefore a language model is computed for relevant and non relevant product items separately.

$P(V_{A,a_i}|rel\_model) = \frac{P(V_{A,a_i})}{\sum_A \sum_{a_i} P(V_{A,a_i})} \forall a_i \in A \forall A \in C, B, Ti$

$P(V_{A,a_i}|nonrel\_model) = \frac{(1-P(V_{A,a_i}))}{\sum_A \sum_{a_i} (1-P(V_{A,a_i}))} \forall a_i \in A \forall A \in C, B, Ti$

Using the click model, the click decision are made for all results in a result page. For the next page of results, the IN will be updated (using *Update model*) and then the relevant and non-relevant models are updated accordingly. Based on a threshold (0.05 is used), the products with the total likelihood score greater than the threshold are clicked in CSUM.

2.6 Interpretation of model parameters The model specific parameters can be used to understand the user behaviour in a session or group of sessions. We present the interpretation of following set of parameters, (1) λ_1, λ_2 indicate the user preference to update to a *Fixed IN* state. The parameters also indicates how likely the user changes his current IN. 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 more time 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. α_{K_1} is the weight for knowledge source based on *Background knowledge* of the product space. If α_{K_1} is high the the user uses popular/frequent words that are used to retrieve the required attribute values. A user exploring also tries to use frequent words in general. A lower value indicates that user has lower *Background knowledge* or does not use this knowledge while querying.

(3) α_{K_2} is the weight for using *Knowledge learnt* of the product space during the session. This parameter can be analysed similar to α_{K_1}, a lower value indicates that the user does not effectively use the *Knowledge learnt* from the product space or the user does not use this knowledge. The *Knowledge learnt* is especially useful for removing words in query reformulation.

(4) α_{K_3} is the weight of *Keyword matching knowledge*. If α_{K_3} is less it is likely that that the user does not the know the direct product space words for his IN and therefore uses other similar or associated words instead of the exact words. The value α_{K_3} is directly proportional to their knowledge of the product space words.

(5) α_{K_4} is the weight of *Linguistic knowledge* or Word similarity based source. This source is used for words that do not belong to the product space. If the user cannot recollect a product space word, they might use the similarity based knowledge source (along with other sources) to come up with similar words. If α_{K_4} is lower implies either the user do not knowledge of similar words or does not choose query words based on word similarity.

3 Learning User Models from Search Logs

The real search sessions can be used to compute estimates of variables like IN and K for simulating a real user or fitting a real search sessions and it can also 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 Walmart E-comm search log.

3.1 Initial State model Given the search task information which is target product T and initial query Q_0 and the search log, the estimation of the initial user state variables is described in the following. Note that in order to fit the CSUM to search sessions we need

to fix some variables because the Information need and Knowledge are defined on large range of values. While fitting the model as described in Section 3.2, we first use the Initial state model to initialize these parameters and then fit the parameters.

Information need initialization

The initial IN is build using the start task information: the target product T, the starting query Q_0 and the background information learned from search logs. Generally, the query indicates only part of the IN of the user and the user can have more product preferences that are latent with respective to query. These preferences can be known based on the final product the user purchases.

The attributes values of target product T are used to make the topic IN θ_T which comprises of $\theta_{T_{fixed}}$, $\theta_{T_{explore}}$ or θ_{T_f}, θ_{T_e} for short. For attributes in T, θ_{T_f} is fixed to 1 and θ_{T_e} to c_0 which is less than 1. The existing user search logs are used to make a background IN θ_b. The common query words used by the users indicate the popular information need (product preferences) of the users like popular brands or title words that are preferred. Therefore, it is likely that the user will have a higher preference for these attribute values in the beginning. The probability of an attribute value among queries of search log is used to make θ_{b_f} and θ_{b_e} where θ_{b_f} only has probabilities belonging to attribute values of T and θ_{b_e} has probabilities belonging to all attribute values. The topic based IN and background based IN are then combined as $(1-\alpha_1)*\theta_T + \alpha_1 * \theta_b$ for both both fixed and explore IN to make $P(V_{A,a_i}|\theta_{Tb_f})$ and $P(V_{A,a_i}|\theta_{Tb_e})$. As θ_T is more important, the value of α_1 is generally very small.

The first query Q_0 provides important information about the initial IN of the user because attributes values corresponding to the query words of Q_0 indicates initial product preferences of the user. Using Q_0, we adjust the $P(V_{A,a_i})$, and estimate $P(f_A)$, $P(e_A)$ for all A. This is equivalent to maximum likelihood solution of IN in order to generate Q_0. For attribute values that are in Q_0, the probabilities are raised to maximum value of all attribute value preferences of that attribute and for other the probabilities remain same, let this adjusted initial IN be denoted $P(V_{A,a_i}|Q_0,T,b_f)$, $P(V_{A,a_i}|Q_0,T,b_e)$. Finally to estimate the $P(f_A)$, $P(e_A)$, we use the a_i in Q_0. $P(f_A)$ is proportional to the likelihood of attributes of Q_0 from $P(V_{A,a_i}|Q_0,T,b_f)$, and similarly for $P(e_A)$ and then they are normalized such that $P(f_A)+P(e_A)=1$.

Therefore, the probability of preference for an attribute value a_i belonging to attribute A can be computed as

$$(P(f_A)*P(V_{A,a_i}|Q_0,T,b_f))+ \\ ((1-P(f_A))*P(V_{A,a_i}|Q_0,T,b_e))$$

Knowledge of the product space

As described in Section 2.2, the *Knowledge learnt* is null in the beginning because the user learns this knowledge during the session while interacting with the system's results. The *Background knowledge* 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 co-occurrence of w_q and *cat* in a user sessions which implies that w_q is among the query words used in session while *cat* is the category attribute of the final product bought in the session and the probability is computed by normalizing it across all *cat*. Here, we assume that all the queries used in a session are relevant to retrieve that category attribute that the user finally purchasing. This assumption is reasonable because in most of the cases, the category attribute is fixed by the user from the beginning and the search results are all from within one category or from very similar categories. Similarly, $P_{kb}(w_p|w_q)$ is computed but with more constraints, we consider only the final interaction in a session before purchase that has query Q_f and the final product P_f. The probability is directly proportional to number of times $w_q \in Q_f$ and $w_p \in P_f$ among the search log sessions and computed by normalized it across all w_p. This *Background knowledge* variable remains same all through the session.

3.2 Fitting using Search logs Using the real search sessions, we can evaluate the model and fit the model to obtain optimal model parameters that explains the real user. By fitting the model to real search log sessions, we evaluate the performance of model description and methods, representation in imitating a real user in a search session and thus its ability to simulate real users. By fitting the model, we also find optimal model parameters that explain the real search session. The proposed model has several parameters, $\{\alpha_1, \alpha_{iupdate}, \alpha_{kupdate}, \{\alpha_{K_i}\}, \lambda_1, \lambda_2\}$. Note that these parameters are same for all attributes. While simulating a user, the specific parameters can be initialized in different ways to obtain variation in session interaction. Some of the parameters ($\{\alpha_1, \alpha_{iupdate}, \alpha_{kupdate}\}$) are considered general (can be assumed to be same for all users and all information needs) while others ($\{\{\alpha_{K_i}\}, \lambda_1, \lambda_2\}$) are specific indicating a specific user behaviour. While fitting the model, the general parameters are learned from all the session whereas the specific parameters are learned separately for a single session or a subset of the sessions with same user or with same or similar information need (same target product).

Estimating the best parameters The best parame-

ters are estimated using query reformulation action and click action. The optimization function tries to maximize the likelihood of correct actions for both query reformulation and clicks and the model parameters are found accordingly. For a query reformulation, let the *true_edits* be the actual words removed or added to the query, the $final\_score(w|IN, \{K_i\})$ of these words should be maximized and the score for other words in top-k words ($top - k\_edits$) should be minimized. Therefore optimization for query reformulation is given by,

$$Of_1 = (\frac{1}{N}\sum_i(\sum_{w\in true\_edits_i}\frac{final\_score(w|IN,\{K_i\})}{|true\_edits_i|} - \sum_{w\in top-k\_edits_i\setminus true\_edits_i}\frac{final\_score(w|IN,\{K_i\})}{|top-k\_edits_i\setminus true\_edits_i|}))$$

where N is the number of reformulations in a session, $true\_edits_i$, $top - k\_edits_i$ are $true\_edits$, $top - k\_edits$ of i^{th} reformulation.

For click action, we consider different measures because it is difficult to predict the exact clicks correctly. Therefore, we consider average pairwise similarity between actual clicks and clicks generated by the model. We used *jaccard similarity* here. Along with similarity, we also compare the number of clicks, the difference between actual number of clicks (*true_clicks*) and the generated number of clicks (*gen_clicks*) should be less. All the clicks in a session are grouped together for computing average similarity an difference in number of clicks. The jaccard similarity $jaccard\_sim$ is computed using appended text of title, category, brand of the product. The optimization function for clicks is,

$$Of_2 = (\frac{1}{|true\_clicks|*|gen\_clicks|}\sum_{t_i}\sum_{g_i} jaccard\_sim(t_i, g_i) + \frac{1}{(1+|true\_clicks|-|gen\_clicks|)})$$

The best specific parameters, $\{params\} = \{\alpha_{K_i}\}, \lambda_1, \lambda_2\}$, are given by

$$\{best\_params\} = argmax_{params}\frac{1}{|S|}\sum_{s\in S}(Of_{1s} + Of_{2s})$$

where S is set of sessions.

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 and the overall performance declines a lot. Therefore, some simplifications are done in order to fit the model to a session. To estimate best parameters, every query reformulation is treated as the first reformulation in the session. The query after the previous reformulation is treated as the first query and the information need is created accordingly. The interactions in between are used to update the IN and Knowledge. Then the query reformulation process is carried out. In this way, the errors made in the previous reformulation and IN update will not carried along to the next reformulation. Similarly for clicks, the actual clicks until the previous set of interactions are considered known and the IN is updated using actual clicks and then the clicks for the current set of results are predicted/generated. Therefore, the mistakes made in previous click decisions will not be effect the next click decisions. The model parameters should be same throughout the session and the score of each query reformulation/click is aggregated to get the performance on a session as given in Of_1, Of_2.

4 Experiments

We explain some implementation details first. For finding values of some general parameters, a validation set of search log sessions is used and the general parameters are set as, $c_0 = 0.5$, $\alpha_1 = 0.3$, $\alpha_{kupdate} = 0.5$, $\alpha_{iupdate} = 0.1$. For simplification, we consider that the user is fixed about category attribute, so $P(f_C) = 1$ for all experiments. All the functions used across the model are computed in following way, in Section 2.3.1 $f(_2) = (4 + 20 * \lambda_2$, $P_o(f_A) = \lambda_1 * \frac{\sum_{o_i\in\{o_i\}} P(V_{A,o_i}|Q_0,t,b_{fixed})}{\sum_{a_i} P(V_{A,a_i}|Q_0,t,b_{fixed})}$, in Section 2.3.2 $f_1(r_p, clk_p) = (\frac{1+4*clk_p}{1+0.2*r_p})$, $f_2(w_p, r_p, clk_p) = (\frac{c(w_p,p)(1+4*clk_p)}{\sum_{w'_p\in p} c(w'_p,p)(1+4*clk_p)})$. Finally, to estimate best specific parameters, 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 generated clicks. We also evaluated on new test sessions by estimated parameters on train sessions with same final target product. From the results we observed that the model performs reasonable well and also generalizes to large set of session, we do not show those results due to space constraints.

4.1 Analysis of User parameters In the following, we verify if the model parameters correlate with the corresponding user behaviour characteristic as described in Section 2.6. We perform this analysis for (λ_1) which indicates exploration behaviour of the user. We choose some observable actions of the users which approximately indicate that the user is exploring. While exploring, the user generally click various other products different from the final product they purchase. The user may click or view the final product in the results but do not purchase it as they are exploring and are not decided. Therefore, we compute the following two estimates using all the clicks of the user in a session. $Exploring\_with\_clicks(Ec) = \frac{\sum(1-jaccard_sim(clicked,final))}{no.of clicked products}$ where $jaccard_sim$ is jaccard similarity. $Exploring\_with\_not\_deciding(EnD)$ is no.of times final product is viewed before purchasing. In

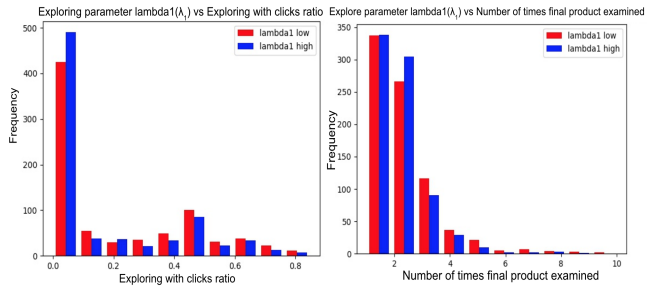

Figure 2: Histogram of Ec, EnD for high and low λ_1

this work we experimented with only discrete values for λ_1 which are $\{0.1, 0.3, 0.5, 0.7\}$. Therefore we analyzed the parameter by dividing it to two groups low,high: low: $0.1, 0.3$, high: $0.5, 0.7$. As λ_1 is inversely proportional to amount of exploration, if λ_1 is low we expect the user is exploring and therefore the Ec, End should be higher. If λ_2 is high we expect the user is not exploring, so Ec, End should be less. Figure 2a, 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, that means it is more frequent that when λ_1 is high when Ec, EnD are low. For higher values of Ec, EnD, λ_2 low (red) dominates which implies that it is more frequent that λ_2 will be low when Ec, EnD are high. Therefore, the histograms agrees with our hypothesis about λ_1, Ec, End and shows that indeed λ_1 indicates the exploration user behaviour.

4.2 Analysis of user behavior using the model

In this section, we show how the proposed model can be used to analyze user behavior in ways that we cannot do with existing models. For example, we can leverage the capability of the model to model the cognitive state of a user and use the estimated parameters of the model based on search log to reveal and understand a user's behavior. Below we will show some preliminary results of behavior analysis. We use the six interpretable parameters from Section 2.6 of the model to obtain meaningful representation of user behaviour in search session: a data point in 6 dimensional space and perform further analysis to understand variation patterns of user behavior.

4.2.1 Variations of user behavior Understanding variations of user behavior is necessary for adapting search service effectively to a specific user in a specific search context. In general, different users searching for the same product would show variations of their behavior, and a user can have different behaviors while searching for different type of products in different categories. Does the behavior vary more across users or across products?

To address this question, we first can compute the variance in user behaviour for same user across different products, $var_u = \frac{1}{|S_u|} * \sum_{s \in S_u} (dist(s, \mu_u)^2)$, where S_u is the set of the sessions of user u, μ_u are the mean point of the sessions S_u, s is the data point of a session. Next, we compute the variance in behaviour of users buying the same product in the end, $var_p = \frac{1}{|S_p|} * \sum_{s \in S_p} (dist(s, \mu_p)^2)$ where S_p is the set of sessions where product p is product purchased in the end. μ_p is the mean point of sessions S_p. We also examine variance along each of the six parameters. Table 1 shows var_u of some users and average var_u of all users. It also shows var_p for some products and average var_p for all products. The table shows the variances along each individual parameters for some users/products and for all users/products. As we consider only the sessions with at least one reformulation, number of sessions for a user is very less with highest being only 12 and the highest number of sessions for a product is only 25. The first rows with "Mean" indicate the average variances for the users/products.

From Table 1, it can be inferred that the variance of parameters of sessions of user and product are in the same range. The user variance is slightly more, for example the variance for a product with 7 different users is around 0.35,0.41 while for the same user but 7 different sessions it is 0.44,0.43. Further, the average variance for all users is 0.311 whereas the average variance for all products is 0.239. Therefore, we conclude that on an average the variance in user behaviour purchasing same product is less than the variance in behaviour of same user buying different products.

By observing individual parameter variances in Table 1, we infer that the variance in λ_1, λ_2 is larger than other parameters $\{\alpha_{K_i}\}$, this implies that most important/frequent difference in behaviours is the exploring or fixed IN behaviour of users. This is also further shown in Section 4.2.2 through common cluster patterns.

Table 1: Variance of *same user sessions* and *same product sessions*. The table shows average of variance using all parameters var_u, var_p average variance along each each parameter.

UserId	$\|S_u\|$	var_u	λ_1	λ_2	α_{K_1})	α_{K_2}	α_{K_3}	α_{K_4}
Mean	-	0.311	0.059	0.064	0.042	0.052	0.047	0.046
1	12	0.426	0.076	0.076	0.059	0.067	0.088	0.06
2	7	0.446	0.088	0.073	0.065	0.069	0.062	0.088
3	7	0.439	0.088	0.088	0.051	0.068	0.088	0.055
ProductId	$\|S_p\|$	var_p	λ_1	λ_2	α_{K_1}	α_{K_2}	α_{K_3}	α_{K_4}
Mean	-	0.239	0.045	0.035	0.046	0.039	0.037	0.039
1	25	0.443	0.089	0.053	0.089	0.053	0.079	0.078
2	19	0.353	0.089	0.089	0.059	0.041	0.018	0.054
3	18	0.410	0.049	0.061	0.085	0.061	0.075	0.078

4.2.2 Common patterns through clustering All the sessions are represented using the 6 model specific

parameters and are clustered. We used K-means clustering. Each cluster center is interpreted as a behaviour pattern representing that cluster. Clustering is done with different number of clusters like 2, 3, 32 and the cluster centers are shown in Table 2. For 32 clusters, only the largest 4 cluster centers are shown in the Table 2. From Table 2, it can be inferred that the difference between cluster centers at just 2 clusters is that one has λ_1 high and λ_2 low and the other has λ_1 low and λ_2 high, the other parameters have same value for both clusters. It implies that the most distinguishable behaviour patterns among E-comm search users are the exploration and fixed Information need (IN) behaviours. With 3 clusters, the next major difference in behaviour patterns is at α_{K_3}, α_{K_4}, the two most different type of knowledge sources used for making queries: the extent to which a user is using only product space wordsK_3 or similar words from outside the product spaceK_4. The cluster center implies that the user uses only either one of the sources more so the values of α_{K_3}, α_{K_4} are inversely correlated. Therefore, the first division of users occurs with type of IN and the second important division is with the types of knowledge sources used.

Table 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

5 Conclusions and Future Work

We proposed a novel cognitive user model CSUM for E-Comm search, which is an interpretable generative model that simulates how a user might behave in 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, re-formulate a query as needed, click on results, with interpretable parameters that can be estimated using a search log.

The model has many applications. One application that we have explored in depth is to use it 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 cognitive states of users, CSUM enables us to reveal interesting user behavior patterns that cannot be extracted using existing models. Our preliminary evaluation shows several interesting findings about user behavior. For example, we show that there is greater variation of a user's behavior across different products than the variation of 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 of user behavior is to what extent the user wants to explore in the search process, 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. Another major application of the model is to enable construction many user simulators based on search log data, which are needed for quantitative evaluation of interactive search engines as well as training reinforcement learning algorithms for optimizing E-Comm search. It would be highly interesting to explore such future applications of CSUM.

References

[1] F. Baskaya, "Simulating Search Sessions in Interactive Information RetrievalEvaluation", PhD thesis, University of Tempere; 2014.

[2] Ben Carterette, Ashraf Bah, and Mustafa Zengin, "Dynamic test collectionsfor retrieval evaluation" In ICTIR; 2015, pp. 91–100.

[3] David Maxwell and Leif Azzopardi, "Agents, simulated users and humans:An analysis of performance and behaviour", In CIKM; 2016, pp. 731–740.

[4] Sondhi, Parikshit and Sharma, Mohit and Kolari, Pranam and Zhai, ChengXiang, "A Taxonomy of Queries for E-commerce Search" In ACM SIGIR; 2018, pp. 1245–1248.

[5] Guo, Qi and Agichtein, Eugene. "Ready to buy or just browsing? Detecting web searcher goals from interaction data.", In ACM SIGIR; 2010, pp. 130–137.

[6] Hassan, Ahmed and White, Ryen W and Dumais, Susan T and Wang, Yi-Min, "Struggling or exploring? Disambiguating long search sessions", In WSDM; 2014, pp. 53–62.

[7] Hassan Awadallah, Ahmed and White, Ryen W and Pantel, Patrick and Dumais, Susan T and Wang, Yi-Min, "Supporting complex search tasks", In CIKM; 2014, pp. 829–838

[8] Wang, Kuansan and Gloy, Nikolas and Li, Xiaolong, "Inferring search behaviors using partially observable Markov (POM) model", In WSDM; 2010, pp. 211–220.