Task-based Recommendation on a Web-Scale

Yongfeng Zhang[†‡], Min Zhang[†], Yiqun Liu[†], Chua Tat-Seng[§], Yi Zhang[‡], Shaoping Ma[†]
[†]Department of Computer Science & Technology, Tsinghua University, Beijing, 100084, China
[‡]School of Engineering, University of California, Santa Cruz, CA 95060, USA
[§]School of Computing, National University of Singapore (NUS), 117417, Singapore
zhangyf07@gmail.com, {z-m,yiqunliu,msp}@tsinghua.edu.cn, dcscts@nus.edu.sg, yiz@soe.ucsc.edu

***Abstract*—The Web today has gone far beyond a tool for simply posting and retrieving information, but a universal platform to accomplish various kinds of tasks in daily life.**

However, research and application of personalized recommendation are still mostly restricted to intra-site vertical recommenders, such as video recommendation in YouTube, or product recommendation in Amazon. Usually, they treat users' historical behaviors as discrete records, extract collaborative relations therein, and provide intra-site homogeneous recommendations, without specific consideration of the underlying tasks that inherently drive users' browsing actions.

In this paper, we propose task-based recommendation to offer cross-site heterogenous item recommendations on a Web-scale, which better meet users' potential demands in a task, e.g., one may turn to Amazon for the dress worn by an actress after watching a video on YouTube, or may turn to car rental websites to rent a car after booking a hotel online. We believe that task-based recommendation would be one of the key components to the next generation of universal web-scale recommendation engines.

Technically, we formalize tasks as demand sequences embedded in user browsing sessions, and extract frequent demand sequences from large scale browser logs recorded by a well known commercial web browser. Based on these demand sequences, we predict the upcoming demand of a user given the current browsing session, and further provide personalized heterogeneous recommendations that meet the predicted demands. Extensive experiments on cross-site heterogenous recommendation with real-world browsing data verified the effectiveness of our framework.

Keywords*—*Task-based recommendation; Task mining; Demand prediction; Collaborative filtering; Browser log analysis

I. Introduction

With the ability to recommend items of potential interests to users, the importance of personalized recommender systems [1] has been recognized in various online applications, such as video recommendation in YouTube, product recommendation in Amazon, and friend recommendation in Facebook. However, the research and application of personalized recommendation techniques are still mostly focused on *vertical* domains, which typically provide intra-site homogeneous recommendations of items within the website.

However, with the fast webifying of our physical world, many of our daily activities that previously can only be conducted offline, are now easily performed online. These include shopping, learning, entertaining, socializing and many business affairs. The Web has been far more than a tool for gathering information, but an integrated platform where a large amount of daily *tasks* can be accomplished, which has fundamentally changed the way people use the Web.

The accomplishment of many tasks require the collaboration of heterogeneous items across different websites. For example, a user who makes travel arrangements for an upcoming holiday may need to buy air tickets from an airline website, book hotels on a hotel reservation website, and further turn to car rental websites for local car rentals. This inherently task-driven consecutive series of actions on multiple websites has highlighted the importance of developing (cross-site) heterogenous recommendation techniques, that help users to discover items of potential needs from a wider scope outside the current single site.

The research on Personalized Recommender Systems (PRS) [1] has long been focusing on Collaborative Filtering (CF)-based [2] and the related techniques, without explicit mining, modeling, and prediction of the underlying tasks and diverse needs that inherently dictate users' browsing behaviors. Here, we argue that a clear understanding of users' potential demands during the browsing process is of fundamental importance, because the expectation of de facto intelligent Web services could hardly be possible if we ignore the inherently coherent task flows of human beings and simply model them as a set of discrete records, especially in personalized recommendation settings.

To bring the research on recommender systems a step closer towards a key component for the next generation of intelligent Web, we propose a heterogeneous task-based recommendation framework. It attempts to predict the upcoming demands of a user by analyzing the historical browsing records of users and the nature of current task. The framework serves as a universal recommendation engine during the browsing process, by providing recommendations on heterogeneous items from multiple websites that meet user's ongoing tasks.

To achieve this goal, one of the key problems is the real-time prediction of user demands. For example, current video sharing websites would provide a recommendation list consisting of other related films within this site for a user who has just finished watching the film *Titanic*, as shown by the top-right part of Figure 1. This vertical recommendation is in fact assuming that users' demand is still to watch more films, which may not be true especially given that he/she has spent hours finishing the previous one. In this case, the system can better meet the potential demand of users with heterogeneous recommendations, such as the product *Ocean Heart Necklace* (clue of the film) from Amazon; the sound track of *My Heart Will Go On* (theme song of the film) from Pandora; or even a related online game from Yahoo! Games, as shown in the bottom-right part of Figure 1.

The task mining and demand prediction at Web-scale has been extremely difficult for an individual website, because of

978-1-4799-9926-2/15/$31.00 ©2015 IEEE

the technical difficulty of collecting users' cross-site browsing behaviours from other websites, which leads to the lack of sufficient data for demand estimation. However, the existence of modern browsers has now made it easy for Web-scale data collection and analysis, and the huge amount of browser logs makes it possible to conduct Web-scale task mining and demand prediction for heterogenous task-based recommendation.

In this work, we treat the users' browsing sessions recorded by browsers as *demand sequences*, where each *demand* is represented by a set of websites of the same category that satisfy a particular user need. For example, Amazon and eBay are for the same demand of *online shopping*, while YouTube is for the demand of *online videos*. Note that the *demand* does not necessarily mean that the user has a targeted *intent* for the follow-on browsing actions, rather, it signifies a kind of potential need that may be of interest to the user.

We conduct frequent demand sequence mining based on a large collection of browsing session records to extract *tasks*, which indicate how users' demands drift during the browsing process. This helps us to predict the follow-on demands of a user given his/her browsing records in the current session. Once the follow-on demands for a user are predicted in the current browsing session, we can further provide task-based recommendations with items that meet with the predicted potential demands of the user.

The rest of the paper is organized as follows: We review related works in Section II, and introduce the problem formalization in Section III. We present our framework for task-based recommendation in Section IV, including task mining, demand prediction, and the construction of personalized heterogenous recommendation lists. Experimental results are presented in Section V, and we conclude this work with some of the future research directions in Section VI.

II. Related work

With the ability to help discover items of potential interests to users, Personalized Recommender Systems (PRS) [3] have been widely integrated into many online applications, such as e-commerce, social networks, and online review services. Early systems for personalized recommendation rely on content-based approaches [4], which make recommendations by analyzing item features or user demographics. Recently, the Collaborative Filtering (CF)-based [2] approaches, especially those based on various Matrix Factorization (MF) [5] techniques, have gained great popularity due to their free from human efforts, superior performance in the precision of rating prediction, and the ability to leverage the wisdom of crowds.

However, the research of personalized recommendation, including the dominating CF-based methods, have long been reluctantly or even intentionally ignoring the explicit mining, modeling, and predicting of user's demands. Typically, they treat users' historical behavior logs as discrete records, and attempt to estimate latent representations in machine learning frameworks for rating prediction [2]. However, it has been shown that the performance on numerical rating prediction is not necessarily related to the performance of recommendation in practical systems [6]. As a result, the explicit consideration of users' potential demands for more informed

Fig. 1. Previous vertical recommender systems provide intra-site homogeneous recommendation results, while heterogeneous task-based recommendation can provide cross-site heterogeneous recommendations that may better meet users' demands.

recommendations may be important in the practical application of recommender systems.

For a relatively long period, and especially in the recent years, the research community of Information Retrieval (IR) has been investigating task-based search [7], [8], [9], [10], [11], which helps users to better reach the targeted information by mining and modeling users' task behaviors. Among the many research efforts for task-based search, one of the key problems is the task-based mining of user intents [12], [13], [14], [15], which states the importance to gain a better understanding of users in IR tasks.

Recently, Wang et al [16] investigated session-aware recommendation in E-commerce. By taking session-level temporal effects into account, recommendations with logical causal relationships can be provided in a session. However, the *sessions* investigated are still within intra-site homogeneous (e-commerce products) purchasing behaviors of users, which differs from the practical *tasks* in our definition that users fulfill on the Web-scale. Zhang et al [17], on the other hand, propose to investigate cross-site recommendation based on Bordered Block Diagonal Form (BBDF) structured matrices, thus to bridge the heterogeneous data from different websites. However, the scalability limitation makes such approaches only able to handle a small number of websites, and they still rely on simple MF-based rating predictions without the consideration of tasks. Besides, both of the approaches model user behaviours on products directly, which inherently avoided the explicit mining of user demands.

To help users discover things of potential interests throughout the process of browsing the Web, we propose task-based recommendation that provides heterogeneous recommendations. Fortunately, the presence of many modern web browsers today has made it technically easy to collect various user browsing behaviors from all aspects [18].

Similar to the research on task-based search [12], [13], [14], task-based recommendation takes the mining and predicting of user demands as a core component, because a correct understanding of the users has long been considered as of crucial importance not only to various online applications, but also to the whole intelligence of the Web [19]. The extraction of users' demand sequence patterns requires Frequent Pattern Mining (FPM) techniques [20], [21], which is a key research topic in the community of data ming.

The research of FPM investigates algorithms for a wide scope of data mining problems, including frequent item set

mining [22], association rule mining [23], frequent sequence mining [24], [25], [26], and frequent tree or graph mining [27], [28]. In this work, we formalize user browsing tasks as demand sequences, and we thus focus on frequent sequence mining algorithms [24] to extract tasks from browser logs, based on which to predict the upcoming demands of a user given his/her browsing records in the current session. Task-based personalized recommendations are thus provided to meet with the predicted demands of the users.

III. Problem Formalization

In this section, we present some of the key concepts and definitions in this work.

Definition 1: A **link** l is the URL of a specific page or item on the Web, which is recorded in the browser logs. A **website** w is the second-level domain of URLs, e.g., music.google.com, developed for a specific user need.

Note that each link l is included in a single website w, which is the second-level domain of l. For example, let $w =$ *www.ebay.com*, then $l =$ *www.ebay.com/itm/131380177014* refers to a product item in the website.

Definition 2: A **demand** $d \in \mathcal{D} = \{d_1, d_2, \cdots, d_m\}$ describes a specific type of user need, where $\mathcal{D}$ is the set of all possible demands that we consider. A demand is identified as a collection of functionally similar websites, i.e., $d_i = \{w_{i1}, w_{i2}, \cdots, w_{in_i}\}$.

We distinguish the demands of different websites by their second-level domains. For example, *music.google.com* is of the same demand (music) as *music.yahoo.com*. However, they are different from the demand (news) of *news.google.com*. We adopt the frequently used web directory from the Open Directory Project[1] (ODP) for website demand classification, and a demand d corresponds to a category in ODP.

Definition 3: We use $l \in w$ to denote that l belongs to website w. We set the demand implied by a link l to be the same as w, i.e., if $l \in w$ and $w \in d$, then $l \in d$.

Definition 4: A **session** s is an ordered browsing sequence recorded by the browser. In this work, we introduce the following three forms in progressive order to describe a session recorded by the web browser:

- $s = \langle w_{s1} w_{s2} \cdots w_{sn_s} \rangle$ is an **initial session** recorded by the browser, where $w_{si} = \langle l_{si}^1 l_{si}^2 \cdots l_{si}^{n_{si}} \rangle$ is a sequence of links that a user browsed consecutively in the same website, as it is usually seen that users tend to browse in the same website for a continuous period of time.
- $s = \langle d_{s1} d_{s2} \cdots d_{sn_s} \rangle$ is a **demand session** deduced from an initial session, where d_{si} is the demand corresponding to w_{si}, i.e., $w_{si} \in d_{si}$.
- Users frequently turn to search engines or navigational pages to find target sites. However, the search engine or navigational pages are out of our consideration for recommendation. We thus remove the search and navigation demands to get a **clear demand session**.

[1] http://www.dmoz.org

In this work, a session is identified from the time the user enters the browser, including both launching the browser and switching to the browser from another application, until the time the user leaves the browser, including both shutting down the browser and switching to another application.

Definition 5: A **task** $t = \langle d_{t1}, d_{t2}, \cdots, d_{tp} \rangle$ is a demand sequence showing the drifting of users' demands when browsing, and the length $p = |t|$ is referred to as its **order**.

Given a database of clear demand sessions $\mathcal{S} = \{s_1, s_2, \cdots, s_n\}$, where $|\mathcal{S}| = n$, $s_i = \langle d_{i1} d_{i2} \cdots d_{in_i} \rangle$; and a minimum support threshold $s_{\min} \in \mathbb{N}, 0 < s_{\min} \leq n$; then the task t is **frequent** if $\sup_{\mathcal{S}}(t) \geq s_{\min}$, where $\sup_{\mathcal{S}}(t)$ is the **support** of task t in $\mathcal{S}$, i.e., $\sup_{\mathcal{S}}(t) = |\{s_i \in \mathcal{S} | t \sqsubseteq s_i\}|$. $t \sqsubseteq s_i$ means that t is a subsequence of s_i, namely, there exist integers $1 \leq j_1 < j_2 < \cdots < j_p \leq n_i$ such that $d_{ij_k} = d_{tk}$.

Definition 6: Given the demand set $\mathcal{D}$, the database of clear demand sessions $\mathcal{S}$, and the minimum support threshold $s_{\min}$, the **task mining** procedure finds the set of frequent demand sequences (i.e., task set) $\mathcal{T}_{\mathcal{S}}(s_{\min}) = \{t = \langle d_{t1} d_{t2} \cdots d_{tp} \rangle | p \geq 2, d_{ti} \in \mathcal{D}, \sup_{\mathcal{S}}(t) \geq s_{\min}\}$.

We adopt the clear demand sessions for task mining, and use the initial sessions to provide personalized recommendations collaboratively once the upcoming demands of a user is predicted based on the extracted tasks. The reason that we avoid conducting CF on the initial sessions directly for recommendation, is the inherent difficulty for effective CF on the rather sparse Web-scale data. Besides, the understanding of users through explicit mining and prediction of users' demands is especially important in the scenario of heterogeneous recommendation. However, we will provide the results of direct CF methods in the experiments for performance comparison.

For easy reference, the notations used in the definitions of this work are summarized in Table I.

IV. Task-based Recommendation

In this section, we present a two-stage framework for task-based recommendation. The system overview of our framework is shown in Figure 2.

In the first stage, we collect large scale browser logs of users and construct the initial session records. Based on the website classification system of ODP, the initial sessions are then transformed to demand sessions, and the task set (i.e., frequent demand sequences) are further extracted from these demand sessions through task mining.

TABLE I. A summary of the notations used.

l	Links that compose the initial browsing sessions
w	Website, represented by its second-level domain
d	Demand, represented by a set of website domains
$\mathcal{D}$	The set of all possible demands, where $\|\mathcal{D}\| = m$
s	Session, i.e., a sequence of links (in an initial session) or demands (in a demand session)
$\mathcal{S}$	The set of session records for task mining, $\|\mathcal{S}\| = n$
t	Task, i.e., a frequent demand sequence, where $p = \|t\|$ is the order of task t
$\sup_{\mathcal{S}}(t)$	The support of task t in session records $\mathcal{S}$
$s_{\min}$	Minimum support, where $s_{\min} \in \mathbb{N}, 0 < s_{\min} \leq n$
$\mathcal{T}_{\mathcal{S}}$	The task set (a set of frequent demand sequences) extracted from $\mathcal{S}$ by task mining

The second stage examines a user's current browsing session. Based on the current browsing session and the task set constructed in stage one, the system predicts the upcoming demands of the user, and provides heterogeneous item recommendations that meet with the predicted demands.

In the followings, we introduce the technical details of our task-based recommendation framework.

A. Collaborative Task Mining

The procedure of task mining presented by Definition 6 generally boils down to a frequent sequence mining problem, which has been investigated extensively by the Data Mining research community [20].

One may refer to Table II for an example of the task mining produce. In this work, we extract not only the closed sequences[2] but all the demand sequences that are frequent in the database of demand sessions. In other words, we include those "shot" tasks (tasks with lower orders) in our extracted task set as long as they are frequent in the demand sessions, no matter whether they are the subsequences of some "longer" tasks. This is to guarantee that we can extract those short but very frequent tasks, as they may exposit clear relations between demands. However, if we restrict ourselves to the mostly adopted closed frequent sequence mining algorithms, these short tasks may be excluded from the final task set by longer but less frequent (yet still frequent enough to satisfy the minimum support threshold) tasks.

TABLE II. AN EXAMPLE OF THE TASK MINING PROCEDURE.

Session records	Session1: **Video,E-commerce,Music**,News,News Session2: Game,**Video,E-commerce,Sports**,Game,Video Session3: Social,**Video,E-commerce,Music**,Social,Blogs	
Extracted tasks	Task1: Video,E-commerce	Order=2,Support=3
	Task2: Video,E-commerce,Music	Order=3,Support=2

According to Definition 5, each element of the sequences in our framework is treated as a single demand rather than a set. As a result, we can easily adopt many of the frequent sequence mining algorithms. For the fact that there exists a unique task set $\mathcal{T}_{\mathcal{S}}$ for the task mining problem given the demand set $\mathcal{D}$, demand session database $\mathcal{S}$, and the minimum support threshold $s_{\min}$, what we are concerned is with the efficiency instead of effectiveness. As a result, we choose the SPAM algorithm[3] [26] for task mining, which is the widely adopted algorithm for sequence mining in data mining.

Based on the different settings of minimum support threshold $s_{\min}$, the number of tasks extracted may be different. In our experimentation, we can obtain $1,273 \sim 11,568$ tasks with $s_{\min}$ ranging from $10,000$ to $1,000$, respectively. The detailed statistics and evaluation results of the extracted task set $\mathcal{T}_{\mathcal{S}}$ will be reported in the experiments.

B. Task-based Demand Prediction

To provide recommendations for the current task of a user, we need to predict the upcoming demands first. To achieve this goal, we estimate the probability of each demand $d \in \mathcal{D}$ being the follow-on demand of the current session.

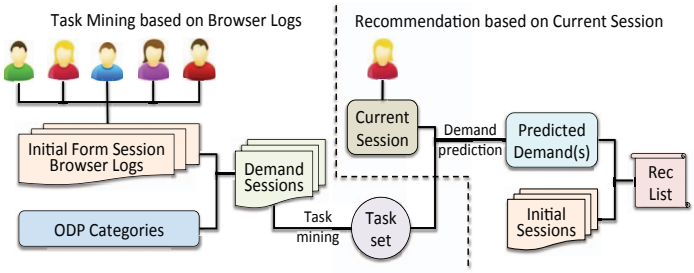


Fig. 2. Framework overview. In Stage one, demand sessions are constructed from browser logs, and the task set is extracted through task mining. In Stage two, the system predicts a user's upcoming demands based on the current session and provide heterogeneous recommendations accordingly.

Suppose the task set $\mathcal{T}_{\mathcal{S}} = \{t_1, t_2, \cdots, t_q\}$, where $q = |\mathcal{T}_{\mathcal{S}}|$ is the total number of tasks. Each task $t_k = \langle d_{k1} d_{k2} \cdots d_{kp_k} \rangle$, where demand $d_{ki} \in \mathcal{D} = \{d_1, d_2, \cdots, d_m\}$, and $p_k = |t_k|$ is the order of t_k. Still, let $\sup_{\mathcal{S}}(t_k)$ be the support of task t_k in the demand session database $\mathcal{S}$, and let $t_c = \langle d_{c1} d_{c2} \cdots d_{cp_c} \rangle$ be the clear demand form of the current browsing session of a user. We then attempt to estimate the probability of each demand in $\mathcal{D}$ given the current browsing session t_c and the extracted task set $\mathcal{T}_{\mathcal{S}}$, namely:

$$\Pr(d_{c(p_c+1)} = d | t_c, \mathcal{T}_{\mathcal{S}}, \sup_{\mathcal{S}}(\cdot)) \text{ for } d \in \mathcal{D} \tag{1}$$

To do so, we first estimate the pair-wise joint probability $\Pr\langle d_i, d_j \rangle$ based on $\mathcal{T}_{\mathcal{S}}$ and $\sup_{\mathcal{S}}(\cdot)$ for each demand pair:

$$\Pr\langle d_i, d_j \rangle = \frac{\sum_{k=1}^{q} \delta_{\langle d_i d_j \rangle \sqsubseteq t_k} \cdot p_k \cdot \sup_{\mathcal{S}}(t_k)}{\sum_{d'_i, d'_j \in \mathcal{S}} \left(\sum_{k=1}^{q} \delta_{\langle d'_i d'_j \rangle \sqsubseteq t_k} \cdot p_k \cdot \sup_{\mathcal{S}}(t_k) \right)} \tag{2}$$

where $\delta_{\langle d_i d_j \rangle \sqsubseteq t_k}$ is an indicator function, whose value is 1 if $\langle d_i d_j \rangle$ is a subsequence of t_k, and 0 otherwise. Note that we use angle brackets to denote that the pairs are ordered and thus we usually have $\Pr\langle d_i, d_j \rangle \neq \Pr\langle d_j, d_i \rangle$ when $i \neq j$.

We consider both the order p_k and the support $\sup_{\mathcal{S}}(t_k)$ in a multiplication manner according to the intuition of the task mining algorithms in Section IV-A, where longer and/or more frequent tasks are more "reliable". Besides, tasks with higher orders have smaller supports compared with its subsequences, and we consider both the order and support to seek a balance.

Given the pair-wise empirical probabilities, we further estimate the probability of the upcoming demand for the current session $t_c = \langle d_{c1} d_{c2} \cdots d_{cp_c} \rangle$:

$$\Pr(d_{c(p_c+1)} = d | t_c, \mathcal{T}_{\mathcal{S}}, \sup_{\mathcal{S}}(\cdot)) = \sum_{r=1}^{p_c} \Pr\langle d_{cr}, d \rangle \tag{3}$$

The demands $d \in \mathcal{D}$ are further ranked in descending order of probability in Eq.(3), and we select the top demands to provide recommendations. For clarity, the procedure of task-based demand prediction is summarized in Algorithm 1.

C. Task-based Recommendation

We consider providing both site-level and link-level recommendations. The site-level recommendation provides recommended sites from the predicted demands, while the link-level recommendation provides specific recommended links

[2] A frequent sequence is closed if it is not a subsequence of any other frequent sequences.

[3] http://himalaya-tools.sourceforge.net/Spam/

(i.e., items in a site). The site-level recommendation acts as a browsing navigator that inspires users' potential demands, while the link-level recommendation displays the actual heterogenous recommendations. Similar to task-based demand prediction, we adopt the initial session records to calculate the probability of each site/link and provide recommendations according to the ranked list.

1) Site-Level Recommendation: Suppose the initial session set is $\mathcal{S}_w = \{s_1, s_2, \cdots, s_{q_w}\}$, where a record is $s_k = \langle w_{k1} w_{k2} \cdots w_{kp_k} \rangle$. Let the current browsing session be $s_c = \langle w_{c1} w_{c2} \cdots w_{cp_c} \rangle$, and let $d \in \mathcal{D}$ be one of the predicted demands (e.g., the top-3 demands) in the previous step of task-based demand prediction. Note that for notational consistency, we still adopt p_k and p_c to denote the order (i.e., the number of sites) of the initial sessions and the current browsing session. We can thus estimate the probability of each site $w \in d$ being the next site of the current session, namely:

$$\Pr(w_{c(p_c+1)} = w | s_c, \mathcal{S}_w) \text{ for } w \in d \tag{4}$$

In the scenario of site-level recommendation during user's browsing process, we only need to care about those site pairs between a site from the current browsing session $w_{cr} \in s_c$ and a site from the predicted demand $w \in d$. Based on statistics from the large amount of initial sessions $\mathcal{S}_w$, the pair-wise probability can be calculated as follows:

$$\Pr\langle w_{cr}, w \rangle = \frac{\sup_{\mathcal{S}_w}(\langle w_{cr} w \rangle)}{|\mathcal{S}_w|} \tag{5}$$

where $\sup_{\mathcal{S}_w}(\cdot)$ is the support of a site sequence in the session records $\mathcal{S}_w$, and $|\mathcal{S}_w|$ is the total number of records in $\mathcal{S}_w$. We do not take the order p_k of an initial session into consideration here as of Eq.(2). This is because each initial session in $\mathcal{S}_w$ itself is only a single and independent record, where there is no frequent sequence mining process to leverage the wisdom of the crowds. As a result, a longer browsing session does not necessarily mean that it is more reliable in practice.

Similar to Eq.(3), the probability of a site $w \in d$ being the upcoming site is estimated as:

$$\Pr(w_{c(p_c+1)} = w | s_c, \mathcal{S}_w) = \sum_{r=1}^{p_c} \Pr\langle w_{cr}, w \rangle \tag{6}$$

We rank the candidate websites $w \in d$ in descending order of probability according to Eq.(6), and select the top-k websites as the recommended sites.

2) Link-level Recommendation: In this section, we still make use of the initial sessions $\mathcal{S}_w = \{s_1, s_2, \cdots, s_{q_w}\}$ as above, where a record is $s_k = \langle w_{k1} w_{k2} \cdots w_{kp_k} \rangle$. However, we further consider the detailed link sequences composing each website in a session, as shown in Definition 4. Here, each website w_{ki} in session s_k is actually a sequence of links that a user browsed consecutively in a website, i.e., $w_{ki} = \langle l_{ki}^1 l_{ki}^2 \cdots l_{ki}^{n_{ki}} \rangle$.

By replacing each website $w_{ki} \in s_k$ with its actual link sequence $w_{ki} = \langle l_{ki}^1 l_{ki}^2 \cdots l_{ki}^{n_{ki}} \rangle$, we denote a session $s_k \in \mathcal{S}_w$ using links directly:

$$s_k = \left\langle \langle l_{k1}^1 l_{k1}^2 \cdots l_{k1}^{n_{k1}} \rangle \langle l_{k2}^1 l_{k2}^2 \cdots l_{k2}^{n_{k2}} \rangle \cdots \langle l_{kp_k}^1 l_{kp_k}^2 \cdots l_{kp_k}^{n_{kp_k}} \rangle \right\rangle \tag{7}$$

Algorithm 1: DEMANDPREDICT$(t_c, \mathcal{D}, \mathcal{T}_\mathcal{S}, \sup_\mathcal{S}(\cdot))$

Input: $t_c := \langle d_{c1} d_{c2} \cdots d_{cp_c} \rangle$ // Current session
$\mathcal{D} := \{d_1, d_2, \cdots, d_m\}$ // Candidate demands
$\mathcal{T}_\mathcal{S} := \{t_1, t_2, \cdots, t_q\}$ // Extracted task set
$\sup_\mathcal{S}(\cdot)$ // Support of each task in $\mathcal{T}_\mathcal{S}$
Output: $\mathcal{L}(t_c)$ // Ranked list of the demands in $\mathcal{D}$

```
P ← 0_{m×m}; // Pair-wise probability matrix init as 0's
Sum ← 0;
for t_k := ⟨d_{k1} d_{k2} ··· d_{kp_k}⟩ ∈ T_S do
    Δ ← 0_{m×m}; // Temporal indicator matrix
    for i ← 1 to p_k − 1 do
        for j ← i + 1 to p_k do
            if Δ[d_ki][d_kj] = 0 then
                P[d_ki][d_kj] ←
                P[d_ki][d_kj] + p_k · sup_S(t_k);
                Sum ← Sum + p_k · sup_S(t_k);
                Δ[d_ki][d_kj] = 1;
P ← P/Sum; // Normalize the elements in P by Sum
P_d ← 0_{1×m}; // Probability of each demand
for d ∈ D do
    for r ← 1 to p_c do
        P_d[d] ← P_d[d] + P[d_cr][d];
L(t_c) ←Rank demands in D in descending order of P_d;
return L(t_c);
```

and similarly, the current session denoted by links is:

$$s_c = \left\langle \langle l_{c1}^1 l_{c1}^2 \cdots l_{c1}^{n_{c1}} \rangle \langle l_{c2}^1 l_{c2}^2 \cdots l_{c2}^{n_{c2}} \rangle \cdots \langle l_{cp_c}^1 l_{cp_c}^2 \cdots l_{cp_c}^{n_{cp_c}} \rangle \right\rangle \tag{8}$$

Based on the large amount of initial sessions $\mathcal{S}_w$, we estimate the pair-wise probability between each link from the current browsing session $l_{cr}^j \in s_c$ and a candidate link $l \in d$, where d is the predicted upcoming demand for the current browsing session, as follows:

$$\Pr\langle l_{cr}^j, l \rangle = \frac{\sup_{\mathcal{S}_w}(\langle l_{cr}^j l \rangle)}{|\mathcal{S}_w|} \tag{9}$$

and the probability of a candidate link $l \in d$ being the upcoming link is summarized as:

$$\Pr(l_{c(p_c+1)}^1 = l | s_c, \mathcal{S}_w) = \sum_{r=1}^{p_c} \sum_{j=1}^{n_{cr}} \Pr\langle l_{c_r}^j, l \rangle \tag{10}$$

We rank the candidate links $l \in d$ according to Eq.(10) and select the top links as the heterogeneous recommendations.

The task-based recommendation framework generally follows a direct top-k recommendation paradigm, because we care more about the recommended items rather than the numerical rating predictions in practical systems. Besides, a superior rating prediction performance (on RMSE for example) does not necessarily mean a good performance on practical top-k recommendation [6]. We will evaluate our task-based recommendation framework in a wide scope on task mining, demand prediction, site-level recommendation, and link-level recommendation in the following section.

TABLE III. THE DEMANDS CONSIDERED IN THIS WORK, AS WELL AS THE PERCENTAGE OF SESSIONS FROM THE TOTAL SESSION RECORDS THAT INCLUDE EACH DEMAND. THOSE DEMANDS WITH PERCENTAGE $\geq 5\%$ ARE BOLDED.

No.	Demmand	%Sessions	No.	Demmand	%Sessions	No.	Demmand	%Sessions	No.	Demmand	%Sessions
1	Life service	4.05%	7	Social culture	0.28%	13	Blogs	**16.30%**	19	Videos & Movies	**9.95%**
2	Economics	2.94%	8	Travelling	0.27%	14	Search engine	**17.79%**	20	Games	**6.73%**
3	Literature	1.68%	9	Politics	0.17%	15	E-commerce	**9.55%**	21	Entertainment	2.42%
4	Education	1.40%	10	Healthcare	0.13%	16	Portals	**7.70%**	22	Social network	**20.30%**
5	News	0.40%	11	Arts	0.05%	17	Navigation	**7.45%**	23	Music	3.28%
6	Sports	0.39%	12	Science	0.03%	18	Computers	3.05%	24	Encyclopedia	4.05%

V. EXPERIMENTS

In this section, we conduct extensive experiments of task-based heterogeneous recommendation, based on the real-world user browsing logs collected by a well-known commercial web browser. We present the experimental setup and the basic statistical analysis of user browsing behaviours first, followed by the evaluations for each component of the task-based recommendation framework.

A. Experimental Setup

We collected the user browsing logs from Jul. 2nd to Aug. 31st in the year of 2013, amounting to 61 days of browsing information for a total of 25,238,600 users.

In accordance with the "long tail" theory, a lot of users in the original logs are inactive users with only a few records, which makes it difficult to learn the personalized preferences for both the task-based and the baseline CF approaches. As such, we extracted a biased sample from the original logs in preference of the active users, so as to focus on the key research problem of personalized task-based recommendation rather than addressing the cold start issues.

We first selected those users that are active in any consecutive period of 7 days (a week). These users are ranked in descending order according to their total PV-values (i.e., the total number of Page Views). This ranking list is then split into 5 equally sized intervals, and we randomly sample a certain percentage of users empirically from each interval. The sampling percentages, number of selected users, and the number of sessions corresponding to the selected users are shown in Table IV.

In the followings, we conduct collaborative task mining on the selected sessions, and perform personalized recommendation for the selected users.

B. Formalizing Demands based on ODP

Based on the definition of demands in Section III, we map the websites in the above session records into a specific demand to construct the demand sessions. Those sites that can not be mapped into any demand are treated as unknown, which are out of consideration in this work. The websites are mapped into 24 demands shown in Table III, where for each demand, we also present the percentage of sessions that include the specific demand. Note that the percentages amount to more than 100%, because a session includes more than one demands.

TABLE IV. STATISTIC OF THE SELECTED USERS AND SESSIONS.

Interval	Percentage	# Users	# Sessions
Top 20%	5/15	62,832	4,838,064
$20\% \sim 40\%$	4/15	50,266	2,714,364
$40\% \sim 60\%$	3/15	37,699	1,357,164
$60\% \sim 80\%$	2/15	25,133	703,724
Last 20%	1/15	12,566	226,188
Total	1	188,496	9,839,504

We see that the demands of "Search engine" and "Navigation" together account for about 25% of the sessions. However, it is practically inappropriate in real-world systems to simply recommend a search engine website or a navigational page to users during browsing process, and we thus exclude these websites in the clear demand sessions.

After pruning away the search engine and navigational websites, the clear demand sessions cover 22 demands, corresponding to $9,359,336$ (about 95%) sessions in the browser logs. In the following sub-sections, we will conduct task mining, demand prediction and task-based recommendation based on these selected session records.

We introduce the comparative algorithms and evaluation metrics for each experimental task separately in the following, because they require different settings.

C. Collaborative Task Mining

We randomly select 80% from the above 9,359,336 sessions for task mining, which amounts to $7,487,468$ clear demand sessions. Given these sessions, the only parameter that one needs to tune for task mining is the minimum support $s_{\min}$, namely, the threshold to determine whether a task is treated as frequent or not. A smaller $s_{\min}$ gives quantitatively more but lower order (thus less reliable) tasks, while a larger $s_{\min}$ value gives quantitatively less but higher order tasks, which are more reliable for their strong support levels. As a result, we experiment with different selections of $s_{\min}$, and trade off between the quality and quantity of the extracted tasks.

We tune the minimum support $s_{\min}$ ranging from 1,000 to 10,000 and record the extracted tasks. Table V shows the number and the average order of the tasks extracted given different choices of minimum supports.

We see that the number of tasks extracted decreases with the increasing of minimum support $s_{\min}$, which is not surprising. Interestingly, the average order of the extracted tasks also decreases with the increasing of $s_{\min}$. On considering the fact that the relatively low-frequency tasks tend to be filtered out when we increase the minimum support threshold, this observation indicates that those high-frequency tasks extracted tend to be lower orders "short" ones.

Figure 4 shows the trends of the total number of extracted tasks and their averaged order given different choices of $s_{\min}$. We also count the number of tasks in each minimum support

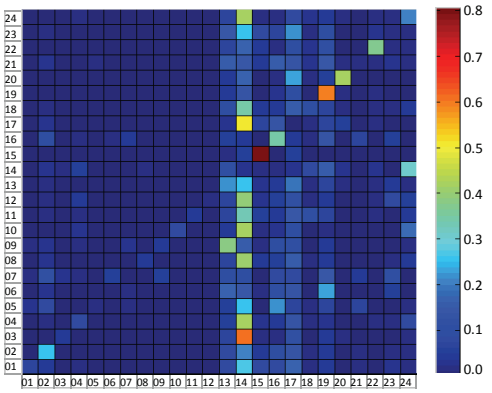


Fig. 3. Heat map of the pair-wise joint probability distribution $\Pr\langle d_i, d_j\rangle$, where d_i is represented by row indices, and d_j represented by column indices.

interval (the bar charts), i.e., the number of tasks that are filtered out when we increase $s_{\min}$ by steps of 1,000. These tasks are those whose support in the session records fall into the corresponding interval. We see that a large amount of tasks fall into the relatively low-frequency support intervals of 1,000 to 3,000, and the high-frequency tasks occupy a small portion in the total amount of tasks. Besides, by taking the averaged order into consideration, we can see that the reliable higher-order tasks constitute a large proportion of the overall tasks, which is an advantage of task extraction based on frequent sequence mining.

However, a dilemma we have to face is that reliable higher-order tasks usually come with lower supports, while those highly supported tasks usually exhibit lower orders. As a result, we have to trade off between the order and support of tasks, which is also the reason that we consider the product of order and support together in Eq.(2). We experiment with different choices of minimum support in the followings for the evaluation of demand prediction and recommendation.

D. Analysis of Pair-wise Probability

An important step for demand prediction is the computation of pair-wise joint probabilities $\Pr\langle d_i, d_j\rangle$ as in Eq.(2), which is an indicator of the pair-wise transition probability between demands. To obtain a clear knowledge of the dependencies between different demands, and also the intuition of user browsing behaviours when surfing online, we conduct detailed investigation on the pair-wise probability based on the extracted tasks.

We first calculate the pair-wise joint probability separately based on each of the ten extracted task sets, which are obtained under different choices of minimum support $s_{\min}$ in the range of $1,000 \sim 10,000$ with steps of 1,000. We then average the ten probability distributions to gain an aggregated view. Figure 3 shows the probability distribution $\Pr\langle d_i, d_j\rangle$ with an intuitional heat map, where the outgoing demands d_i are represented by rows, and the incoming demands d_j by columns. For better understandings, we also include the demand of Search (#14) and Navigation (#17) in Figure 3, although we do not consider them for demand prediction.

One can see that column #14 (corresponding to the demand of Search) is brighter compared with other demands, which exhibits a broadly higher probability of transition from other

TABLE V. TOTAL NUMBER OF EXTRACTED TASKS WITH DIFFERENT CHOICES OF MINIMUM SUPPORT FOR TASK MINING.

$s_{\min}$	1,000	2,000	3,000	4,000	5,000
# of Tasks	11,568	8,583	6,402	3,823	3,365
Average order	5.57	4.94	4.51	3.77	3.63
$s_{\min}$	6,000	7,000	8,000	9,000	10,000
# of Tasks	2,881	2,206	1,764	1,422	1,273
Average order	3.48	3.30	3.19	3.12	3.10

demands to Search. This is because users turn to search engines frequently to locate the demanded sites.

The anti-diagonal elements also exhibit higher transition probabilities, which means a high probability of intra-demand transition during browsing process. This indicates that users usually have to browse across different websites of the same demand to accomplish a specific target in a session. For example, the brightest anti-diagonal element #15 (E-commerce) implies that users may browse across different E-commerce websites to find the appropriate product to buy.

Besides, the many off-diagonal bright elements indicate frequent cross-demand transitions, e.g., the high transit probability from #6 (Sports) to #19 (Videos & Movies) implies that users may turn to sport videos after learning about a match or a player from the sport websites. By taking advantage of the wisdom of crowds from the large amount of browser logs, we are able to predict users' potential demands and provide opportune task-based recommendations.

E. Task-based Demand Prediction

We adopt the remaining 20% of sessions (1,871,868) for demand prediction based on the tasks extracted in the previous experimental section. For each clear demand session in the testing set, we adopt all but the last demand to construct a top-k demand prediction list, and compare the predicted demands with the last one (treated as the groundtruth) to evaluate the performance of demand prediction.

We adopt the evaluation measures of Success@k and NDCG@k, where the latter takes the position of the target demand into consideration. Let $N = 1,871,868$ denote the total number of sessions for testing, and let n be the number of sessions where the last demand is successfully recommended

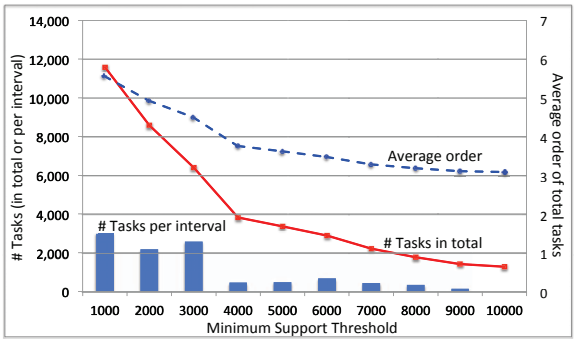


Fig. 4. Left axis: The number of tasks (red solid line) and the number of tasks in each support interval (blue bars). Right axis: Average order under different choices of minimum support (blue dashed line).

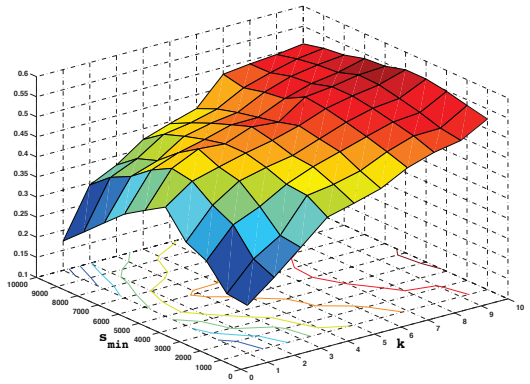


Fig. 5. Evaluation results on NDCG under different $(k, s_{\min})$ pairs.

in the top-k demand prediction list, then:

$$\text{Success@}k = \frac{n}{N}, \text{NDCG@}k = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\text{IDCG}}\sum_{j=1}^{k}\frac{2^{\delta_{ij}}-1}{\log_2(j+1)} \tag{11}$$

where $\delta_{ij} = 1$ if the j-th demand in the prediction list for session i is the target demand, and 0 otherwise.

To the best of our knowledge, this is the first time to investigate the problem of task-based recommendation, and there is no direct baseline method in this problem setting. However, we adopt some of the most commonly used baselines, and also adapt the most state-of-the-art approaches in the research of collaborative filtering and context-aware recommendation to conduct extensive performance comparison. Note that we exclude the demands of Search and Navigation and predict the demand of a user within the remaining 22 demands.

Random: We select k demands randomly as a prediction list, where the sampling weight for each demand is in proportion to its percentage of sessions in Table III.

MostPopular: For each session, we construct the demand prediction list with the top-k most frequent demands according to Table III.

MFCF: The Matrix Factorization (MF) based Collaborative Filtering (CF) approach. We construct the user-demand relation matrix where each element of the matrix is the sigmoid normalized count that a user browsed a demand. We adopt the frequently used MF approach in [29] for matrix completion, and select the top-k demands of a user as the prediction list for his/her session in the test set.

Tensor: The commonly used tensor factorization approach for context-aware recommendation [30]. We consider $|\mathcal{D}|$ layers as the context dimension of the tensor $\mathbb{T}$, where $|\mathcal{D}| = 24$ in this experiment. Each context layer $\mathbb{T}[d]$ is a $|\mathcal{U}|\times|\mathcal{D}|$ matrix corresponding to a specific demand, where $|\mathcal{U}|$ is the number of users. The element $\mathbb{T}[d_i][u][d_j]$ is the count that demand pair $\langle d_i d_j\rangle$ occurs in the demand sessions of user u. After tensor completion and normalization, we actually obtain the pairwise transition probability $\Pr\langle d_i, d_j\rangle$ for each user u. This is used to substitute our task-based transition probability in Eq.(2), while the following procedure of demand prediction is the same.

One can see that the Random and MostPopular methods are non-personalized and independent from the current browsing session of a user, while the MFCF and Tensor approaches are personalized and able to adjust the predictions with respect to users' current browsing sessions.

We set the number of latent factors to 20 for MFCF and Tensor, and adopt the Stochastic Gradient Descent (SGD) algorithm for model learning. We temporally fix the minimum support $s_{\min} = 5,000$ for our task-based approach, and experiment with the length k of the prediction list. The results on Success@k and NDCG@k with different choices of k are shown in Figure 6, and Table VI shows the specific values under the common choices of k.

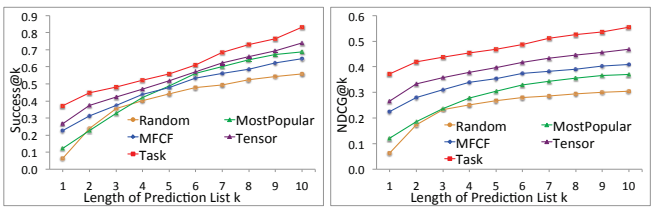
TABLE VI. SPECIFIC VALUES OF SUCCESS AND NDCG GIVEN SELECTED CHOICES OF PREDICTION LENGTH k.

Metric	Success@k				NDCG@k			
k	1	3	5	10	1	3	5	10
Random	0.062	0.357	0.442	0.557	0.062	0.233	0.267	0.305
MostPopular	0.121	0.327	0.488	0.689	0.121	0.238	0.304	0.370
MFCF	0.225	0.375	0.478	0.648	0.225	0.312	0.354	0.409
Tensor	0.266	0.423	0.509	0.738	0.266	0.359	0.397	0.469
Task	**0.372**	**0.483**	**0.558**	**0.832**	**0.372**	**0.457**	**0.487**	**0.575**

Generally, the Success and NDCG increase with the number of demand predictions that we provide, and our Task-based approach outperforms the comparative methods consistently under different settings of the length of prediction list. The reported improvements are significant at the level of 0.05 or better due to large size of our dataset. Specifically, our task-based approach is able to predict the upcoming demand correctly for 37.2% of the sessions with only a single prediction among the 22 potential demands.

We further tune the the minimum support $s_{\min}$ from 1,000 to 10,000 at steps of 1,000 under each prediction length k, and the evaluation results on NDCG are shown in Figure 5. We see that the NDCG@k still rises with k given a fixed $s_{\min}$, but interestingly we find that when we tune $s_{\min}$ given a fixed k, the NDCG rises to a maxima first and then drops.

This observation conforms with our analysis of task mining in Section V-C. When a smaller threshold $s_{\min}$ is used, we can extract more tasks for the pair-wise probability estimation. However, this may also introduce less reliable tasks due to their lower support in session database. On the other hand, when larger values of $s_{\min}$ are adopted, we can extract more reliable tasks but this also results in insufficient number of tasks for accurate probability estimation. As a result, we have to trade off between the quality and quantity of the extracted tasks in practice. The experimental results suggest an optimal selection of $s_{\min}$ at around $5,000$, and this is the reason we fix $s_{\min} = 5,000$ in the previous experiment.



(a) Success@k (b) NDCG@k

Fig. 6. The evaluation results on Success and NDCG v.s. the length of demand prediction list k.

We can also see that the effect of $s_{\min}$ becomes less obvious and the marginal curve tends to be more stable under higher values of k. This is because we can always make a correct guess given a sufficiently high number of chances.

F. Task-based Recommendation

We still leverage the 20% (1,871,868) sessions, and adopt the demands predicted for each session in the previous experiment to conduct and evaluate both the site-level and link-level recommendation. We set $s_{\min}$ to the optimal value of 5,000 in this experiment.

For site-level recommendation, we reserve the last website of a testing session (i.e., the current browsing session) for evaluation, and predict the reserved website based on the previous sites in this session. For each of the k predicted demands in the previous experiment, we construct the ranked list of websites in that demand according to Eq.(6), and adopt the top site as the site-level prediction for that demand. The k predicted demands thus offer us a site-level recommendation list of length k.

Similarly, we reserve the last link of each session, and rank the links for each of the k predicted demands according to Eq.(10), and construct the link-level recommendation list by selecting the top returned link for each demand.

We still compare with the previous baseline methods:

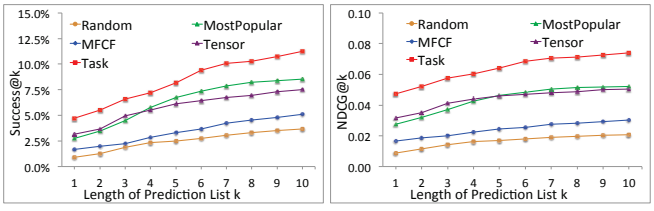
Random: Sample a site/link for each demand randomly by probability in proportion to the percentage of a site/link.

MostPopular: Select the most popular site/link for each predicted demand.

MFCF: We construct the user-site and user-link relation matrices of sigmoid normalized browsing counts, and still adopt the MF approach in [29]. We select the site/link with the highest predicted rating for each demand.

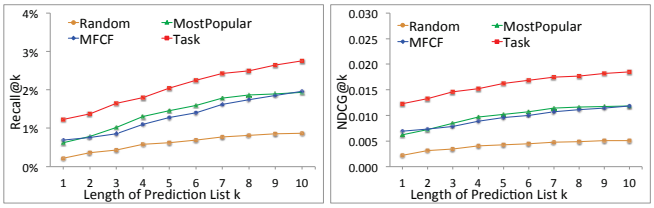
Tensor: Similar to the experiment on demand prediction, we construct the site-level tensor and estimate the pairwise probability between sites to substitute Eq.(5) for context-aware site-level recommendation. Due to the extremely large amount of links in the dataset, it is computationally impossible for link-level tensor factorization in a common experimental setting. As a result, we do not report the link-level performance of the tensor factorization approach.

For easy comparison, we also adopt Success@k and NDCG@k in Eq.(11) for evaluation. Figure 7 shows the results for site-level recommendation, and Figure 8 presents the link-level recommendation results.



(a) Success@k (Site-level) (b) NDCG@k (Site-level)

Fig. 7. Evaluation on Success and NDCG v.s. prediction length k for Site-level recommendation.



(a) Success@k (Link-level) (b) NDCG@k (Link-level)

Fig. 8. Evaluation on Success and NDCG v.s. prediction length k for Link-level recommendation.

The experimental results show that our task-based recommendation gives consistently better performance on both Success and NDCG for site-level and link-level recommendations (significant level at 1% or better). The site-level absolute success ranges from $4.72\% \sim 11.26\%$ and NDCG from $0.047 \sim 0.075$ given k from 1 to 10, correspondingly. For link-level recommendation, the success ranges from $1.23\% \sim 2.75\%$ and NDCG from $0.012 \sim 0.019$. This is an exciting performance for commercial browser-based applications because only a single percentage point of improvement on prediction could bring us a huge increment on profits due to the large amount of users and items.

Besides, it is surprising to find that a simple most popular approach can be comparable to or even better than MF-based collaborative filtering and context-aware tensor factorization. The underlying reason can be the inherent sparsity of site- or link-level relations between users and the large amount of items from the Web-scale data, and thus it can be difficult for CF approaches to estimate accurate user-item relationships. However, our task-based approach is able to extract more reliable relations between users and items based on frequent task mining. This gives us relations of higher confidence, and can thus be more suitable for the problem of Web-scale recommendation.

G. Recommendation within the Right Demand

One may see that the previous experiment on task-based recommendation relies on the performance of demand prediction, because it would be impossible to recommend the accurate site/link given the wrong demand. To eliminate the effect of wrong demand predictions, and to gain a more informed understanding of the performance of task-based recommendation, we further conduct site- and link-level recommendation given the right follow-on demand of a session.

To achieve this goal, we determine rather than predict the demand of a testing session using the last site or link directly, and construct the site- and link-level recommendation list within this demand, according to Eq.(6) and Eq.(10), respectively. We select the top-k sites/links from the ranked list for that demand as the final site-level or link-level recommendations directly. The evaluation results for task-based recommendation under this setting are shown in Table VII.

We see that given the right demand, the Success gains an absolute increment of $3\% \sim 8\%$ on site-level, and $2\% \sim 4\%$ on link-level recommendation (significant level at 1% or better), which verifies that the task-based recommendation component relies on the performance of demand prediction. As a result, the performance of task-based site-level and link-level recommendation can well be further improved with the

TABLE VII. SUCCESS AND NDCG GIVEN THE RIGHT DEMAND.

Metric		Success@k			NDCG@k		
k		1	5	10	1	5	10
Right Demand	Site	0.075	0.144	0.193	0.075	0.109	0.124
	Link	0.033	0.056	0.064	0.033	0.044	0.047
Predicted Demand	Site	0.047	0.082	0.113	0.047	0.064	0.074
	Link	0.012	0.021	0.028	0.012	0.016	0.019

continuous development of more accurate and meticulously designed demand prediction techniques.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we propose task-based recommendation to provide heterogenous recommendations during users' browsing process at Web-scale. To achieve this goal, we leverage large scale user browsing logs collected by a real-world commercial browser, and investigate on demand-, site-, and link-levels. We extract tasks based on frequent sequence mining from browser logs, and further predict the demand of a user given the current browsing session. Based on the predicted demands, we are able to provide both site-level and link-level recommendations. Extensive experimental results not only verified the effectiveness of our framework, but also gave us an intuitional understanding of the characteristics of users' browsing behaviour when surfing online.

We believe that the task-based recommendation has the potential to shape the future of universal recommendation engines for the upcoming era of intelligent Web, and there is much to be done on this emerging topic. Aside from the problem formalization in this work, it will be exciting to see other potential formalizations of task-based recommendation under different application scenarios. Besides the sequential relations adopted in this work, we can investigate other possible relationships between demands, based on sets, hierarchical trees, or even graphs, etc. We can also construct more general and flexible demands based on clustering or topic modeling instead of the direct ODP-based approach. Moreover, we can even predict a series of follow-on demands beyond the next single one for the current browsing session, and adapt various machine learning algorithms for further improvements on the performance of task mining, demand prediction, and task-based recommendation.

ACKNOWLEDGEMENT

This work was supported by National Key Basic Research Program (2015CB358700) and Natural Science Foundation (61472206) of China. Part of this work was conducted at the Tsinghua-NUS NExT Search Centre, which is supported by the Singapore National Research Foundation & Interactive Digital Media R&D Program Office, MDA under research grant (WBS:R-252-300-001-490).

REFERENCES

[1] P. B. Kantor, L. Rokach, F. Ricci, and B. Shapira, *Recommender Systems Handbook*. Springer, 2011.

[2] X. Su and T. M. Khoshgoftaar, "A Survey of Collaborative Filtering Techniques," *Advances in AI*, 2009.

[3] J. Liu, M. Z. Q. Chen, J. Chen, F. Deng, H. Zhang, Z. Zhang, and T. Zhou, "Recent Advances in Personal Recommender Systems," *Jour. Info. Sys. Sci.*, 2009.

[4] M. J. Pazzani and D. Billsus, "Content-Based Recommendation Systems," *The Adaptive Web LNCS*, pp. 325–341, 2007.

[5] G. Takacs, I. Pilaszy, B. Nemeth, and D. Tikk, "Investigation of Various Matrix Factorization Methods for Large Recommender Systems," *Proc. ICDM*, 2008.

[6] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of Recommender Algorithms on Top-N Recommendation Tasks," *RecSys*, pp. 39–46, 2010.

[7] P. Vakkari, "Task-based Information Searching," *Annual Review of Info. Sci. & Tech.*, vol. 37, no. 1, pp. 413–464, 2003.

[8] C. Lucchese, S. Orlando, R. Perego, F. Silvestri, and G. Tolomei, "Identifying Task-based Sessions in Search Engine Query Logs," *WSDM*, pp. 277–286, 2011.

[9] D. Kelly, J. Arguello, and R. Capra, "NSF Workshop on Task-based Information Search Systems," *SIGIR*, 2013.

[10] A. Kotov, P. N. Bennett, R. W. White, S. T. Dumais, and J. Teevan, "Modeling and Analysis of Cross-Session Search Tasks," *SIGIR*, pp. 5–14, 2011.

[11] R. W. White, W. Chu, A. Hassan, X. He, Y. Song, and H. Wang, "Enhancing Personalized Search by Mining and Modeling Task Behavior," *WWW*, pp. 1411–1420, 2013.

[12] A. Ashkan and C. L. A. Clarke, "Impact of Query Intent and Search Context on Click-through Behavior in Sponsored Search," *Knowledge and information systems*, vol. 34, no. 2, pp. 425–452, 2013.

[13] Z. Cheng, B. Gao, and T.-Y. Liu, "Actively Predicting Diverse Search Intent from User Browsing Behaviors," *WWW*, pp. 221–230, 2010.

[14] J. Arguello, F. Diaz, J. Callan, and J.-F. Crespo, "Sources of Evidence for Vertical Selection," *SIGIR*, 2009.

[15] R. Mehrotra, E. Yilmaz, and M. Verma, "Task-Based User Modelling for Personalization via Probabilistic Matrix Factorization," *RecSys*, 2014.

[16] J. Wang, "Session Aware Recommender System In E-Commerce," *PhD Dissertation in UCSC*, 2013.

[17] Y. Zhang, "Browser-oriented Universal Cross-Site Recommendation and Explanation based on User Browsing Logs," *RecSys*, pp. 433–436, 2014.

[18] R. West, R. W. White, and E. Horvitz, "From Cookies to Cooks: Insights on Dietary Patterns via Analysis of Web Usage Logs," *WWW*, pp. 1399–1410, 2013.

[19] D. A. Norman and S. W. Draper, *User Centred System Design: New Perspectives on Human-Computer Interaction*. L. Erlbaum Associates Inc., Hillsdale, NJ, 1986.

[20] J. Han, H. Cheng, D. Xin, and X. Yan, "Frequent Pattern Mining: Current Status and Future Directions," *Data Mining and Knowledge Discovery*, vol. 15, no. 1, pp. 55–86, 2007.

[21] C. C. Aggarwal and J. Han, *Frequent Pattern Mining*. Springer, 2014.

[22] C. Borgelt, "Frequent Item Set Mining," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 2, no. 6, pp. 437–456, 2012.

[23] S. Kotsiantis and D. Kanellopoulos, "Association Rules Mining: A Recent Overview," *Transactions on Computer Science and Engineering*, vol. 32, pp. 71–82, 2006.

[24] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu, "PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth," *IEEE Computer Society*, 2001.

[25] J. Wang and J. Han, "BIDE: Efficient Mining of Frequent Closed Sequences," *ICDE*, pp. 79–90, 2004.

[26] J. Ayres, J. Gehrke, T. Yiu, , and J. Flannick, "Sequential Pattern Mining using A Bitmap Representation," *KDD*, pp. 429–435, 2002.

[27] M. J. Zaki, "Efficiently Mining Frequent Trees in a Forest," *KDD*, pp. 71–80, 2002.

[28] X. Yan and J. Han, "CloseGraph: Mining Closed Frequent Graph Patterns," *KDD*, pp. 286–295, 2003.

[29] Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[30] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver, "Multiverse Recommendation: N-dimensional Tensor Factorization for Context-aware Collaborative Filtering," *RecSys*, pp. 79–86, 2010.