# Extracting Hierarchies of Search Tasks & Subtasks via a Bayesian Nonparametric Approach

Rishabh Mehrotra[†] and Emine Yilmaz[†*]
[†]University College London, London, United Kingdom
[*]The Alan Turing Institute, British Library, London, United Kingdom
{r.mehrotra,e.yilmaz}@cs.ucl.ac.uk

## ABSTRACT

A significant amount of search queries originate from some real world information need or tasks [13]. In order to improve the search experience of the end users, it is important to have accurate representations of tasks. As a result, significant amount of research has been devoted to extracting proper representations of tasks in order to enable search systems to help users complete their tasks, as well as providing the end user with better query suggestions [9], for better recommendations [41], for satisfaction prediction [36] and for improved personalization in terms of tasks [24, 38]. Most existing task extraction methodologies focus on representing tasks as flat structures. However, tasks often tend to have multiple subtasks associated with them and a more naturalistic representation of tasks would be in terms of a hierarchy, where each task can be composed of multiple (sub)tasks. To this end, we propose an efficient Bayesian nonparametric model for extracting hierarchies of such tasks & subtasks. We evaluate our method based on real world query log data both through quantitative and crowdsourced experiments and highlight the importance of considering task/subtask hierarchies.

## KEYWORDS

search tasks; bayesian non-parametrics; hierarchical model

**ACM Reference format:**
Rishabh Mehrotra[†] and Emine Yilmaz[†*]. 2017. Extracting Hierarchies of Search Tasks & Subtasks via a Bayesian Nonparametric Approach. In *Proceedings of SIGIR '17, August 07-11, 2017, Shinjuku, Tokyo, Japan, ,* 11 pages.
DOI: 10.1145/3077136.3080823

## 1 INTRODUCTION

The need for search often arises from a person's need to achieve a goal, or a task such as booking travels, buying a house, etc., which would lead to search processes that are often lengthy, iterative, and are characterized by distinct stages and shifting goals. [13]. Thus, identifying and representing these tasks properly is highly important for devising search systems that can help end users complete their tasks. It has previously been shown that these task representations can be used to provide users with better query suggestions [9], offer improved personalization [24, 38], provide better recommendations [41], help in satisfaction prediction [36] and search result re-ranking. Moreover, accurate representations of tasks could also be highly useful in aptly placing the user in the task-subtask space to contextually target the user in terms of better recommendations and advertisements, developing task specific ranking of documents, and developing task based evaluation metrics to model user satisfaction. Given the wide range of applications these tasks representations can be used for, significant amount of research has been devoted to task extraction and representation [12, 13, 15, 17, 21].

Task extraction is quite a challenging problem as search engines can be used to achieve very different tasks, and each task can be defined at different levels of granularity. A major limitation in existing task-extraction methods lies in their treatment of search tasks as flat structure-less clusters which inherently lack insights about the presence or demarcation of subtasks associated with individual search tasks. In reality, often search tasks tend to be hierarchical in nature. For example, a search task like planning a wedding involves subtasks like searching for dresses, browsing different hairstyles, looking for invitation card templates, finding planners, among others. Each of these subtasks (1) could themselves be composed of multiple subtasks, and (2) would warrant issuing different queries by users to accomplish them. Hence, in order to obtain more accurate representations of tasks, new methodologies for constructing hierarchies of tasks are needed.

As part of the proposed research, we consider the challenge of extracting hierarchies of search tasks and their associated subtasks from a search log given just the log data without the need of any manual annotation of any sort. In a recent poster we showed that Bayesian nonparametrics have the potential to extract a hierarchical representation of tasks [25]; we extend this model further to form more accurate representations of tasks.

We present an efficient Bayesian nonparametric model for discovering hierarchies and propose a tree based nonparametric model to discover this rich hierarchical structure of tasks/subtasks embedded in search logs. Most existing hierarchical clustering techniques result in binary tree structures with each node decomposed into two child nodes. Given that a complex task could be composed of an arbitrary number of subtasks, these techniques cannot directly be used to construct accurate representations of tasks. In contrast, our model is capable of identifying task structures that can be composed of an arbitrary number of children. We make use of a number of evaluation methodologies to evaluate the efficacy of the proposed task extraction methodology, including quantitative and qualitative analyses along with crowdsourced judgment studies


*SIGIR '17, August 07-11, 2017, Shinjuku, Tokyo, Japan*

DOI: 10.1145/3077136.3080823

specifically catered to evaluating the quality of the extracted task hierarchies. We contend that the techniques presented expand the scope for better recommendations and search personalization and opens up new avenues for recommendations specifically targeting users based on the tasks they involve in.

## 2 RELATED WORK

Web search logs provide explicit clues about the information seeking behavior of users and have been extensively studied to improve search experiences of users. We cover several areas of related work and discuss how our work relates to and extends prior work.

### 2.1 Task Extraction

There has been a large body of work focused on the problem of segmenting and organizing query logs into semantically coherent structures. Many such methods use the idea of a *timeout* cutoff between queries, where two consecutive queries are considered as two different sessions or tasks if the time interval between them exceeds a certain threshold [6, 10, 33]. Often a 30-minute timeout is used to segment sessions. However, experimental results of these methods indicate that the timeouts are of limited utility in predicting whether two queries belong to the same task, and unsuitable for identifying session boundaries.

More recent studies suggest that users often seek to complete multiple search tasks within a single search session [20, 23] with over 50% of search sessions having more than 2 tasks [23]. At the same time, certain tasks require significantly more effort, time and sessions to complete with almost 60% of complex information gathering tasks continued across sessions [1, 22]. There have been attempts to extract in-session tasks [13, 20, 35], and cross-session tasks [15, 37] from query sequences based on classification and clustering methods, as well as supporting users in accomplishing these tasks [9]. Prior work on identifying search-tasks focuses on task extraction from search sessions with the objective of segmenting a search session into disjoint sets of queries where each set represents a different task [12, 21].

Kotov et al. [15] and Agichtein et al. [1] studied the problem of cross-session task extraction via binary same-task classification, and found different types of tasks demonstrate different life spans. While such task extraction methods are good at linking a new query to an on-going task, often these query links form long chains which result in a task cluster containing queries from many potentially different tasks. With the realization that sessions are not enough to represent tasks, recent work has started exploring cross-section task extraction, which often results in complex non-homogeneous clusters of queries solving a number of related yet different tasks. Unfortunately, pairwise predictions alone cannot generate the partition of tasks efficiently and even with post-processing, the final task partitions obtained are not expressive enough to demarcate subtasks [18]. Finally, authors in [17] model query temporal patterns using a special class of point process called Hawkes processes, and combine topic model with Hawkes processes for simultaneously identifying and labeling search tasks.

Jones et al. [13] was the first work to consider the fact that there may be multiple subtasks associated with a user's information need and that these subtasks could be interleaved across different sessions. However, their method only focuses on the queries submitted by a single user and attempts to segment them based on whether they fall under the same information need. Hence, they only consider solving the task boundary identification and same task identification problem and cannot be used directly for task extraction. Our work alleviates the same user assumption and considers queries across different users for task extraction. Finally, in a recent poster [25], we proposed the idea of extracting task hierarchies and presented a basic tree extraction algorithm. Our current work extends the preliminary model in a number of dimensions including novel model of query affinities and task coherence based pruning strategy, which we observe gives substantial improvement in results. Unlike past work, we also present detailed derivation and evaluation of the extracted hierarchy and application on task extraction.

### 2.2 Supporting Complex Search Tasks

There has been a significant amount of work on task continuation assistance [1, 28], building task tours and trails [30, 34], query suggestions [2, 14, 27], predicting next search action [5] and notes taking when accomplishing complex tasks [8]. The quality of most of these methods depends on forming accurate representations of tasks, which is the problem we are addressing in this paper.

### 2.3 Hierarchical Models

Rich hierarchies are common in data across many domains, hence quite a few hierarchical clustering techniques have been proposed. The traditional methods for hierarchically clustering data are bottom-up agglomerative algorithms. Probabilistic methods of learning hierarchies have also been proposed [3, 19] along with hierarchical clustering based methods [7, 11]. Most algorithms for hierarchical clustering construct binary tree representations of data, where leaf nodes correspond to data points and internal nodes correspond to clusters. There are several limitations to existing hierarchy construction algorithms. The algorithms provide no guide to choosing the correct number of clusters or the level at which to prune the tree. It is often difficult to know which distance metric to choose. Additionally and more importantly, restriction of the hypothesis space to binary trees alone is undesirable in many situations - indeed, a task can have any number of subtasks, not necessarily two. Past work has also considered constructing task-specific taxonomies from document collections [39], browsing hierarchy construction [40], generating hierarchical summaries [16]. While most of these techniques work in supervised settings on document collections, our work instead focused on short text queries and offers an unsupervised method of constructing task hierarchies.

Finally, Bayesian Rose Trees and their extensions have been proposed [3, 4, 32] to model arbitrary branching trees. These algorithms naively cast relationships between objects as binary (0-1) associations while the query-query relationships in general are much richer in content and structure.

We consider a number of such existing methods as baselines and the various advantages of the proposed approach is highlighted in the evaluation section wherein the proposed approach in addition to being more expressive, performs better than state-of-the-art task extraction and hierarchical methods.

| Symbol | Description |
|---|---|
| $n_T$ | number of children of tree T |
| $ab\|c$ | partition of set $\{a,b,c\}$ into disjoint sets $\{a,b\},\{c\}$ |
| ch(T) | children of T |
| $\phi(T)$ | partition of tree T |
| $p(D_m\|T_m)$ | likelihood of data $D_m$ given the tree $T_m$ |
| $\pi_{T_m}$ | mixing proportions of partition of tree $T$ |
| $f(D_m)$ | marginal probability of the data $D_m$ |
| $\mathbb{H}(T)$ | set of all partitions of queries $Q = leaves(T)$ |
| $f(Q)$ | task affinity function for set of queries Q |
| $r^k_{q_i,q_j}$ | the k-th inter-query affinity between $q_i$ & $q_j$ |

**Table 1: Table of symbols**

## 3 DEFINING SEARCH TASKS

Jones et al. [13] was one of the first papers to point out the importance of task representations, where they defined a search task as:

*Definition 3.1.* A *search task* is an atomic information need resulting in one or more queries.

Ahmed et al. [9] later extended this definition to a more generic one, which can also capture task structures that could possibly consist of related subtasks, each of which could be complex tasks themselves or may finally split down into simpler tasks or atomic informational needs. Following Ahmed *et al.* [9], a complex search task can then be defined as:

*Definition 3.2.* A *complex search task* is a multi-aspect or a multi-step information need consisting of a set of related subtasks, each of which might recursively be complex.

The definition of complex tasks is much more generic, and captures all possible search tasks, that can be either complex or atomic (non-complex). Throughout this paper we adopt the definition provided in Definition 3.2 as the definition for a search task.

Hence, by definition a search task has a hierarchical nature, where each task can consist of an arbitrary number of, possibly complex subtasks. An effective task extraction system should be capable of accurately identifying and representing such hierarchical structures.

## 4 CONSTRUCTING TASK HIERARCHIES

While hierarchical clustering are widely used for clustering, they construct binary trees which may not be the best model to describe data's intrinsic structure in many applications, for example, the task-subtask structure in our case. To remedy this, multi-branch trees are developed. Currently there are few algorithms which generate multi-branch hierarchies. Blundel *et al.* [3, 4] adopt a simple, deterministic, agglomerative approach called BRTs (Bayesian Rose Trees) for constructing multi-branch hierarchies. In this work, we adapt BRT as a basic algorithm and extend it for constructing task hierarchies. We next describe the major steps of BRT approach.

### 4.1 Bayesian Rose Trees

BRTs [3, 4] are based on a greedy probabilistic agglomerative approach to construct multi-branch hierarchies. In the beginning,

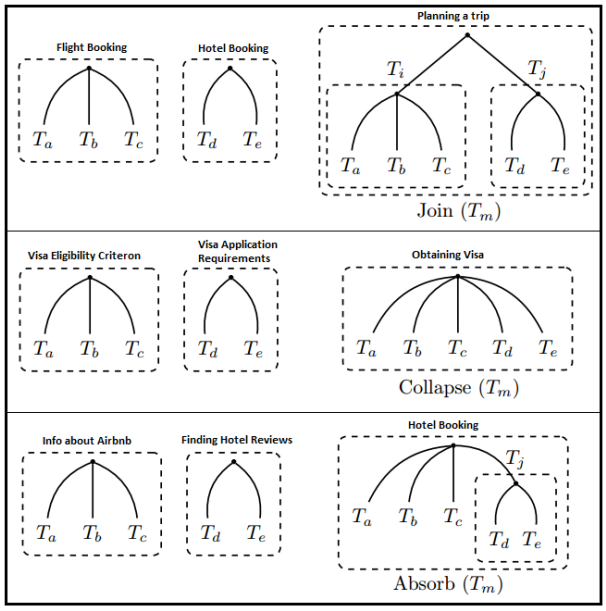


**Figure 1: The different ways of merging trees which allows us to obtain tree structures which best explain the task-subtask structure.**

each data point is regarded as a tree on its own: $T_i = \{x_i\}$ where $x_i$ is the feature vector of i-th data. For each step, the algorithm selects two trees $T_i$ and $T_j$ and merges them into a new tree $T_m$. Unlike binary hierarchical clustering, BRT uses three possible merging operations, as shown in Figure 1:

- **Join**: $T_m = T_i, T_j$, such that the tree $T_m$ has two children now
- **Absorb**: $T_m = children(T_i) \cup T_j$, i.e., the children of one tree gets absorbed into the other tree forming an absorbed tree with >2 children
- **Collapse**: $T_m = children(T_i) \cup children(T_j)$, all the children of both the subtrees get combined together at the same level.

Specifically, in each step, the algorithm greedily finds two trees $T_i$ and $T_j$ to merge which maximize the ratio of probability:

$$\frac{p(D_m|T_m)}{p(D_i|T_i)p(D_j|T_j)} \tag{1}$$

where $p(D_m|T_m)$ is the likelihood of data $D_m$ given the tree $T_m$, $D_m$ is all the leaf data of $T_m$, and $D_m = D_i \cup D_j$ . The probability $p(D_m|T_m)$ is recursively defined on the children of $T_m$:

$$p(D_m|T_m) = \pi_{T_m} f(D_m) + (1 - \pi_{T_m}) \prod_{T_i \in ch(T_m)} p(D_i|T_i) \tag{2}$$

where $f(D_m)$ is the marginal probability of the data $D_m$ and $\pi_{T_m}$ is the "*mixing proportion*". Intuitively, $\pi_{T_m}$ is the prior probability that all the data in $T_m$ is kept in one cluster instead of partitioned into sub-trees. In BRT[4], $\pi_{T_m}$ is defined as:

$$\pi_{T_m} = 1 - (1 - \gamma)^{n_{T_m} - 1} \tag{3}$$

where $n_{T_m}$ is the number of children of $T_m$, and $0 \geq \gamma \leq 1$ is the hyperparameter to control the model. A larger $\gamma$ leads to coarser

| Query-Term Based Affinity ($r^1$) | |
|---|---|
| cosine | cosine similarity between the term sets of the queries |
| edit | norm edit distance between query strings |
| Jac | Jaccard coeff between the term sets of the queries |
| Term | proportion of common terms between the queries |
| **URL Based Affinity ($r^2$)** | |
| Min-edit-U | Minimum edit distance between all URL pairs from the queries |
| Avg-edit-U | Average edit distance between all URL pairs from the queries |
| Jac-U-min | Minimum Jaccard coefficient between all URL pairs from the queries |
| Jac-U-avg | Average Jaccard coefficient between all URL pairs from the queries |
| **Session/User Based Affinity ($r^3$)** | |
| Same-U | if the two queries belong to the same user |
| Same-S | if the two queries belong to the same session |
| **Embedding Based Affinity ($r^4$)** | |
| Embedding | cosine distance between embedding vectors of the two queries |

**Table 2: Query-Query Affinities.**

partitions and a smaller $\gamma$ leads to finer partitions. Table 1 provides an overview of notations & symbols used throughout the paper.

## 4.2 Building Task Hierarchies

We next describe our task hierarchy construction approach built on top of Bayesian Rose Trees. A tree node in our setting is comprised of a group of queries which potentially compose a search task, i.e. these are the set of queries that people tend to issue in order to achieve the task represented in the tree node.

We define the task-subtask hierarchy recursively: T is a task if either T contains all the queries at its node (an atomic search task) or if T splits into children trees as $T = \{T_1, T_2, ..., T_{n_T}\}$ where each of the children trees ($T_i$) are disjoint set of queries corresponding to the $n_T$ subtasks associated with task $T$. This allows us to consider trees as a nested collection of sets of queries defining our task-subtask hierarchical relation.

To form nested hierarchies, we first need to model the query data. This corresponds to defining the marginal distribution of the data $f(D_m)$ as defined in Equation 2. The marginal distribution of the query data ($f(D_m)$) helps us encapsulate insights about task level interdependencies among queries, which aid in constructing better task representations. The original BRT approach [4] assumes that the data can be modeled by a set of binary features that follow the Bernoulli distribution. In other words, features (that represent the relationship/similarities between data points) are not weighted and can only be binary. Binary (0/1) relationships are too simplistic to model inter-query relationships; as a result, this major assumption fails to capture the semantic relationships between queries and is not suited for modeling query-task relations. To this end, we propose a novel query affinity model and to alleviate the binary feature assumption imposed by BRT, we propose a conjugate model of query affinities, which we describe next.

## 4.3 Conjugate Model of Query Affinities

A tree node in our setting is comprised of a group of queries which *potentially* belong to the same search task. The likelihood of a tree should encapsulate information about the different relationships which exists between queries. Our goal here is to make use of the rich information associated with queries and their result set available to compute the likelihood of a set of queries to belong to the same task. In order to do so, we propose a query affinity model which makes use of a number of different inter-query affinities to determine the tree likelihood function.

We next describe the technique used to compute four broad categories of inter-query affinity and later describe the Gamma-Poisson conjugate model which makes use of these affinities to compute the marginal distribution of the data.

**Query-term based Affinity ($r^1$):**
Search queries catering to the same or similar informational needs tend to have similar query terms. We make use of this insight and capture query level affinities between a pair of queries. We make use of cosine similarity between the query term sets, the normalized edit distances between queries and the Jaccard Coefficient between query term sets.

**URL-based Affinity ($r^2$):**
Users tackling similar tasks tend to issue queries (possibly different) which return similar URLs, thus encoding the URL level similarity between pairs of queries into the query affinity model helps in capturing another task-specific similarity between queries. Any query pair having high URL level similarity increase the possibility of the query pair originating from similar informational needs. We capture a number of URL-based signals including minimum and average edit distances between URL domains and jaccard coefficient between URLs.

**User/Session based Affinity ($r^3$):**
It is often the case that users issue related queries within a session so as to satisfy their informational need. We leverage this insight by making use of session level information (as a 0/1 binary feature) and user-level information (as a 0/1 binary feature) in our affinity model to identify queries issued in the same session and by the

same user accordingly.

**Query Embedding based Affinity ($r^4$):**
Word embeddings capture lexico-semantic regularities in language, such that words with similar syntactic and semantic properties are found to be close to each other in the embedding space. We leverage this insight and propose a query-query affinity metric based on such embeddings. We train a skip-gram word embeddings model where a query term is used as an input to a log-linear classifier with continuous projection layer and words within a certain window before and after the words are predicted. To obtain a query's vector representation, we average the vector representations of each of its query terms and compute the cosine similarity between two queries' vector representations to quantify the embedding based affinity ($r^4$).

Table 2 summarizes all features considered to compute these affinities. Our goal is to capture information from all four affinities when defining the likelihood of the tree. We assume that the global affinity among a group of queries can be decomposed into a product of independent terms, each of which represent one of the four affinities from the query-group. For each query group $Q$, we take the normalized sum of the affinities from all pairs of queries in the group $Q$ to form each of the affinity component ($r^k$, k=1,2,3,4).

Poisson models have been shown as effective query generation models for information retrieval tasks [26]. While these affinities could be used with a lot of distributions, in the interest of computational efficiency and to avoid approximate solutions, our model will use a hierarchical Gamma-Poisson distribution to encode the query-query affinities. We incorporate the gamma-Poisson conjugate distribution in our model under the assumptions that the query affinities are discretized and for a group of queries $Q$, the affinities can be decomposed to a product of independent terms, each of which represents contributions from the four different affinity types. Finally, for a tree ($T_m$) consisting of the data ($D_m$), i.e. the set of queries $Q$, we define the marginal likelihood as:

$$f(D_m) = f(Q) = \prod_{k=1}^{k=4} p\left( \sum_{i \in 1 \cdots |Q|} \sum_{j \in 1 \cdots |Q|} r^k_{q_i,q_j} | \alpha_k, \beta_k \right) \quad (4)$$

where $\alpha_k$ & $\beta_k$ are respectively the shape parameter & the rate parameter of the four different affinities. Making use of the Poisson-Gamma conjugacy, the probability term in the above product can be written as:

$$p(r|\alpha,\beta) = \int_\lambda p(r|\lambda)p(\lambda|\alpha,\beta)d\lambda \quad (5)$$

$$= \left\{ \frac{\Gamma(\alpha+r)}{r!\,\Gamma(\alpha)} \left( \frac{\beta}{\beta+1} \right)^\alpha \left( \frac{1}{\beta+1} \right)^r \right\} \quad (6)$$

where $\lambda$ is the Poisson mean rate parameter which gets eliminated from computations because of the Gamma-Poisson conjugacy and where $r$, $\alpha$ & $\beta$ get replaced by affinity class specific values.

## 4.4 Task Coherence based Pruning

The search task extraction algorithm described above provides us a way of constructing a task hierarchy wherein as we go down the tree, nodes comprising of complex multi-aspect tasks split up to provide finer tasks which ideally should model user's fine grained information needs. One key problem with the hierarchy construction algorithm is the continuous splitting of nodes which results in singleton queries occupying the leave nodes. While splitting of nodes which represent complex tasks is important, the nodes representing simple search task queries corresponding to atomic informational needs should not be further split into children nodes. Our goal in this section is to provide a way of quantifying the task complexity of a particular node so as to prevent splitting up nodes representing atomic search task into further subsets of query nodes.

*4.4.1 Identifying Atomic Tasks.* We wish to identify nodes capturing search subtasks which represent atomic informational need. In order to do so, we introduce the notion of *Task Coherence*:

*Definition 4.1. Task Coherence* is a measure indicating the atomicity of the information need associated with the task. It is captured by the semantic closeness of the queries associated with the task.

By measuring Task Coherence, we intend to capture the semantic variability of queries within this task in an attempt to identify how complex or atomic a task is. For example, a tree node corresponding to a complex task like planning a vacation would involve queries from varied informational needs including flights, hotels, getaways, etc; while a tree node corresponding to a finer task representing an atomic informational need like finding discount coupons would involve less varied queries - all of which would be about discount coupons. Traditional research in topic modelling has looked into automatic evaluation of topic coherence [29] via Pointwise Mutual Information. We leverage the same insights to capture task coherence.

*4.4.2 Pointwise Mutual Information.* PMI has been studied variously in the context of collocation extraction [31] and is one measure of the statistical independence of observing two words in close proximity. We wish to compute PMI scores for each node of the tree. A tree node in our discussion so far has been represented by a collection of search queries. We split queries into terms and obtain a set of terms corresponding to each node, and calculate a node's PMI scores using the node's set of query terms.

More specifically, the PMI of a given pair of query terms ($w_1$ & $w_2$) is given by:

$$PMI(w_1,w_2) = log \frac{p(w_1,w_2)}{p(w_1)p(w_2)} \quad (7)$$

where the probabilities are determined from the empirical statistics of some full standard collection. We employ the AOL log query set for this and treat two query terms as co-occurring if both terms occur in the same session. For a given task node ($Q$), we measure task coherence as the average of PMI scores for all pairs of the search terms associated with the task node:

$$PMI - Score(Q) = \frac{1}{|w|} \sum_{i=1}^{|w|} \sum_{j=1}^{|w|} PMI(w_i,w_j) \quad (8)$$

where $|w|$ represents the total number of unique search terms associated with task node $Q$. The node's PMI-Score is used as the final measure of task coherence for the task represented via the corresponding node.

*4.4.3 Tree Pruning.* We use the task coherence score associated with each node of the task hierarchy constructed, and prune lower level nodes of the tree to avoid aggressive node splitting. The overall motivation here is to avoid splitting nodes which represent simple search tasks associated with atomic informational needs. We scan through all levels of the search task hierarchy obtained by the algorithm described above and for each node compute its task coherence score. If the task coherence score exceeds a specific threshold, it implies that all the queries in this particular node are aimed at solving the same or very similar informational need and hence, we prune off the sub-tree rooted at this particular node and ignore all further splits of this node.

## 4.5 Algorithmic Overview

We summarize the overall algorithm to construct the hierarchy by outlining the steps. The problem is treated as one of greedy model selection: each tree T is a different model, and we wish to find the model that best explains the search log data in terms of task-subtask structure.

**Step 1: Forrest Initialization**:
The tree is built in a bottom-up greedy agglomerative fashion, starting from a forest consisting of n (=$|Q|$) trivial trees, each corresponding to exactly one vertex. The algorithm maintains a forest F of trees, the likelihood $p(i) = p(D_i|T_i)$ of each tree $T_i \in F$ and the different query affinities. Each iteration then merges two of the trees in the forest. At each iteration, each vertex in the network is a leaf of exactly one tree in the forest. At each iteration a pair of trees in the forest F is chosen to be merged, resulting in forest $F*$.

**Step 2: Merging Trees**:
At each iteration, the best potential merge, say of trees X and Y resulting in tree I, is picked off the heap. Binary trees do not fit into representing search tasks since a task is likely to be composed of more than two subtasks. As a result, following [3] we consider three possible mergers of two trees $T_i$ and $T_j$ into $T_m$. $T_m$ may be formed by joining $T_i$ and $T_j$ together using a new node, giving $T_m = \{T_i, T_j\}$. Alternatively $T_m$ may be formed by absorbing $T_i$ as a child of $T_j$, yielding $T_m = \{T_j\} \bigcup ch(T_i)$, or vice-versa, $T_m = \{T_i\} \bigcup ch(T_j)$. We explain the different possible merge operations in Figure 1. We obtain arbitrary shaped sub-trees (without restricting to binary tress) which are better at representing the varied task-subtask structures as observed in search logs with the structures themselves learnt from log data. Such expressive nature of our approach differentiates it from traditional agglomerative clustering approaches which necessarily result in binary trees.

**Step 3: Model Selection**:
Which pair of trees to merge, and how to merge these trees, is determined by considering which pair and type of merger yields the largest Bayes factor improvement over the current model. If the trees $T_i$ and $T_j$ are merged to form the tree M, then the Bayes factor score is:

$$SCORE(M; I, J) = \frac{p(D_M|F*)}{p(D_M|F)} \tag{9}$$

$$= \frac{p(D_M|M)}{p(D_i|T_i)p(D_j|T_j)} \tag{10}$$

where $p(D_i|T_i)$ and $p(D_j|T_j)$ are given by the dynamic programming equation mentioned above. After a successful merge, the statistics associated with the new tree are updated. Finally, potential mergers of the new tree with other trees in the forest are considered and added onto the heap.

The algorithm finishes when no further merging results in improvement in the Bayes Factor score. Note that the Bayes factor score is based on data local to the merge - i.e., by considering the probability of the connectivity data only among the leaves of the newly merged tree. This permits efficient local computations and makes the assumption that local community structure should depend only on the local connectivity structure.

**Step 4: Tree Pruning**:
After constructing the entire hierarchy, we perform the post-hoc tree pruning procedure described in Section 4.4 wherein we identify atomic task nodes via their task coherence estimates and prune all child nodes of the identified atomic nodes.

# 5 EXPERIMENTAL EVALUATION

We perform a number of experiments to evaluate the proposed task-subtask extraction method. First, we compare its performance with existing state-of-the-art task extraction systems on a manually labelled ground-truth dataset and report superior performance (5.1). Second, we perform a detailed crowd-sourced evaluation of extracted tasks and additionally validate the hierarchy using human labeled judgments (5.2). Third, we show a direct application of the extracted tasks by using the task hierarchy constructed for term prediction (5.3).

**Parameter Setting:**
Unless stated otherwise, we made use of the best performing hyperparameters for the baselines as reported by the authors. The query affinities in the proposed approach were computed from the specific query collection used in the dataset used for each of the three experiments reported below. While hyperparmeter optimization is beyond the scope of this work, we experimented with a range of the shape and inverse scale hyperparameters ($\alpha$, $\beta$) used for the Poison Gamma conjugate model and used the ones which performed best on the validation set for the search task identification results reported in the next section. Additionally, for the tree pruning threshold, we empirically found that a threshold of 0.8 gave the best performance on our toy hierarchies, and was used for all future experiments.

## 5.1 Search Task Identification

To justify the effectiveness of the proposed model in identifying search tasks in query logs, we employ a commonly used AOL data subset with search tasks annotated which is a standard test dataset for evaluating task extraction systems. We used the task extraction dataset as provided by Lucchese *et al.*[20]. The dataset comprises of a sample of 1000 user sessions for which human assessors were asked to manually identify the optimal task-based query sessions, thus producing a ground-truth that can be used for evaluating

automatic task-based session discovery methods. For further details on the dataset and the dataset access links, readers are directed to Lucchese *et al.*[20].

We compare our performance with a number of search task identification approaches:

- **Bestlink-SVM** [37]: This method identified search task using a semi-supervised clustering model based on the latent structural SVM framework.
- **QC-HTC/QC-WCC** [20]: This series of methods viewed search task identification as the problem of best approximating the manually annotated tasks, and proposed both clustering and heuristic algorithms to solve the problem.
- **LDA-Hawkes** [17]: a probabilistic method for identifying and labeling search tasks that model query temporal patterns using a special class of point process called Hawkes processes, and combine topic model with Hawkes processes for simultaneously identifying and labeling search tasks.
- **LDA Time-Window(TW)**: This model assumes queries belong to the same search task only if they lie in a fixed or flexible time window, and uses LDA to cluster queries into topics based on the query co-occurrences within the same time window. We tested time windows of various sizes and report results on the best performing window size.

*5.1.1 Metrics.* A commonly used evaluation metric for search task extraction is the pairwise F-measure computed based on pairwise precision/recall [13, 15] defined as,

$$p_{pair} = \frac{\Sigma_{i \leq j} \delta(y(q_i), y(q_j)) \delta(\hat{y}(q_i), \hat{y}(q_j))}{\delta(\hat{y}(q_i), \hat{y}(q_j))} \tag{11}$$

$$r_{pair} = \frac{\Sigma_{i \leq j} \delta(y(q_i), y(q_j)) \delta(\hat{y}(q_i), \hat{y}(q_j))}{\delta(y(q_i), y(q_j))} \tag{12}$$

where $p_{pair}$ evaluates how many pairs of queries predicted in the same task, i.e., $\delta(\hat{y}(q_i), \hat{y}(q_j)) = 1$, are actually annotated as in the same task, i.e., $\delta(y(q_i), y(q_j)) = 1$ and $r_{pair}$ evaluates how many pairs annotated as in the same task are recovered by the algorithm. Thus, globally F-measure evaluates the extent to which a task contains only queries of a particular annotated task and all queries of that task. Given $p_{pair}$ and $r_{pair}$, the F-measure is computed as:$F_1 = \frac{2 \times p_{pair} \times r_{pair}}{p_{pair} + r_{pair}}$.

*5.1.2 Results & Discussion.* Figure 2 compares the proposed model with alternative probabilistic models and state-of-the-art task identification approaches by F1 score. To make fair comparisons, we consider the last level of the pruned tree constructed as task clusters when computing pairwise precision/recall values. It is important to note that the labelled dataset has only flat tasks extracted on a per user basis; as a result, this dataset is not ideal for making fair comparisons of the proposed hierarchy extraction method with baselines. Nevertheless, the proposed approach manages to outperform existing task extraction baselines while having much greater expressive powers and providing the subdivision of tasks into subtasks. LDA-TW performs the worst since its assumptions on query relationship within the same search task are too strong. The advantage over QC-HTC and QC-WCC demonstrates that appropriate usage of query affinity information can even better reflect the semantic relationship between queries, rather than exploiting it in some collaborative knowledge.

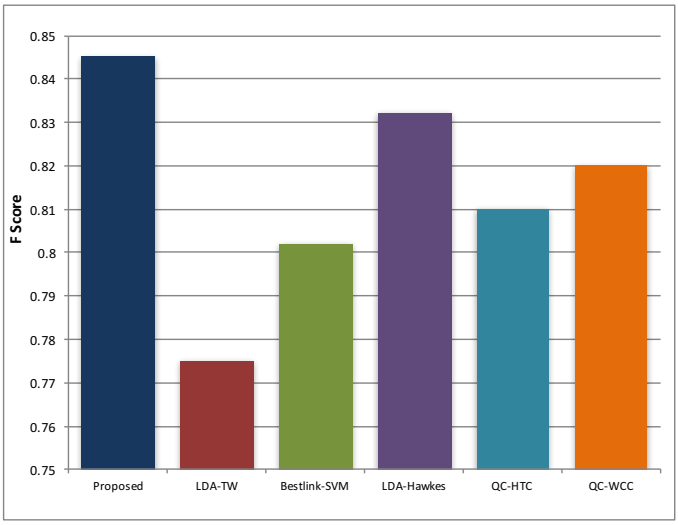


**Figure 2: F1 score results on AOL tagged dataset**

| | Subtask Validity | | | |
|---|---|---|---|---|
| | **Proposed** | **Jones** | **BHCD** | **BAC** |
| Valid | **81%*** | 69% | 51% | 49% |
| Somewhat Valid | 8% | 19% | 17% | 21% |
| Not Valid | 11% | 12% | 32% | 30% |
| | **Subtask Usefulness** | | | |
| Useful | **67%*** | 52% | 41% | 43% |
| Somewhat Useful | 8% | 17% | 19% | 20% |
| Not Useful | 25% | 31% | 40% | 37% |

**Table 4: Performance on Subtask Validity and Subtask Usefulness. Results highlighted with * signify statistically significant difference between the proposed framework and best performing baseline using $\chi^2$ test with $p \leq 0.05$.**

## 5.2 Evaluating the Hierarchy

While there are no gold standard datasets for evaluating hierarchies of tasks, we performed crowd-sourced assessments to assess the performance of our hierarchy extraction method. We separately evaluated the coherence and quality of the extracted hierarchies via two different set of judgements obtained via crowdsourcing.

***Evaluation Setup***

For the judgment study, we make use of the AOL search logs and randomly sampled entire query history of frequent users who had more than 1000 search queries. The AOL log is a very large and long-term collection consisting of about 20 million of Web queries issued by more than 657000 users over 3 months. We run the task extraction algorithms on the entire set of queries of the sampled users and collect judgments to assess the quality of the tasks extracted.

| | Task Relatedness | | | | |
|---|---|---|---|---|---|
| | **Proposed** | **LDA-TW** | **QC-WCC** | **LDA-Hawkes** | **QC-HTC** |
| Task Related | **72%*** | 47% | 60% | 67% | 61% |
| Somewhat Related | 20% | 14% | 15% | 13% | 5% |
| Unrelated | **10%** | 23% | 25% | 20% | 34% |

**Table 3: Performance on Task Relatedness. The results highlighted with * signify statistically significant difference between the proposed approach and best performing baseline using $\chi^2$ test with $p \leq 0.05$.**

Judgments were provided by over 40 judges who were recruited from the Amazon Mechanical Turk crowdsourcing service. We restricted annotators to those based in the US because the logs came from searchers based in the US. We also used hidden quality control questions to filter out poor-quality judges. The judges were provided with detailed guidelines describing the notion of search tasks and subtasks and were provided with several examples to help them better understand the judgement task.

***Evaluating Task Coherence***

In the first study, we evaluated the quality of the tasks extracted by the task extraction algorithms. In an ideal task extraction system, all the queries belonging to the same task cluster should ideally belong to the same task and hence have better task coherence. To this end, we evaluate the task coherence property of the tasks extracted by the different algorithms. For each of the baselines and the proposed algorithm, we select a task at random from the set of tasks extracted and randomly pick up two queries from the selected task. We then ask the human judges the following question:

**RQ1: Task Relatedness:** Are the given pairs of queries related to the same task? The possible options include (i) Task Related, (ii) Somewhat Task Related and (iii) Unrelated.

The task relatedness score provides an estimate of how coherent tasks are. Indeed, a task cluster containing queries from different tasks would score less in Task Relatedness score since if the task cluster is impure, there is a greater chance that the 2 randomly picked queries belong to different tasks and hence get judged Unrelated.

***Evaluating the hierarchy***

While there are no gold standard dataset to evaluate hierarchies, in our second crowd-sourced judgment study, we evaluate the quality of the hierarchy extracted. A valid task-subtask hierarchy would have the parent task representing a higher level task with its children tasks representing more focused subtasks, each of which help the user achieve the overall task identified by the parent task.

We evaluate the correctness of the hierarchy by validating parent-child task-subtask relationships. More specifically, we randomly select a parent node from the hierarchy and then randomly select a child node from the set of its immediate child nodes. Given such parent-child node pairs, we randomly pick 5 queries from the parent node and randomly pick 2 queries from the child node. We then show the human judges these parent and child queries and ask the following questions:

**RQ2: Subtask Validity:** Consider the set of queries representing the search task and the pair of queries representing the subtask. How valid is this subtask given the overall task?

The possible judge options include (i) Valid Subtask, (ii) Somewhat valid and (iii) Invalid. Answering this question helps us in analyzing the correctness of the parent-child task-subtask pairs.

**RQ3: Subtask Usefulness:** Consider the set of queries representing the search task and the pair of queries representing the subtask. Is the subtask useful in completing the overall search task?

The possible judge options include (i) Useful, (ii) Somewhat Useful and (iii) Not Useful. This helps us in evaluating the usefulness of task-subtask pairs by finding the proportion of subtasks which help users in completing the overall task described by the parent node. Overall, the RQ2 and RQ3 help in evaluating the correctness and usefulness of the hierarchy extracted.

***Baselines***

Since RQ1 evaluates task coherence without any notion of task-subtask structure, we compare against the top performing baselines from the task extraction setup described in section 5.1. On the other hand, RQ2 & RQ3 help in answering questions about the quality of hierarchy constructed. To make fair comparisons while evaluating the hierarchies, we introduce additional hierarchy extraction baselines:

- **Jones Hierarchies** [13]: A supervised learning approach for task boundary detection and same task identification. We train the classifier using the supervised Lucchese AOL task dataset and use it to extract tasks on the current dataset used in the judgment study.
- **BHCD** [3]: A state-of-the-art bayesian hierarchical community detection algorithm based on stochastic blockmodels and makes use of Beta-Bernoulli conjugate priors to define a network. We build a network of queries and apply BHCD algorithm to extract hierarchies of query communities.
- **Bayesian Agglomerative Clustering (BAC)** [11]: A standard agglomerative hierarchical clustering model based on Dirichlet process mixtures.

***Results & Discussion***

For the first judgment study, each HIT is composed of 20 query pairs per approach being judged for task relatedness. We had three judges

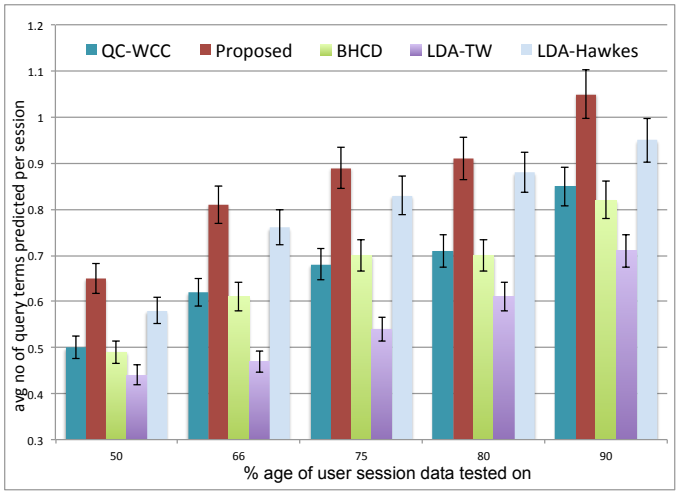


**Figure 3: Term Prediction performance**

work on every HIT. Overall, per method we obtained judgments for 60 query pairs to evaluate the performance on task-relatedness. From among the three judges judging each query-pair, we followed majority voting mechanism to finalize the label for the instance. Table 3 presents the proportions of query pairs judged as related. About 72% of query pairs were judged task-related for the proposed approach with LDA-Hawkes performing second best with 67%. Task relatedness measures how pure the task clusters obtained are, a higher score indicates that the queries belonging to the same task are indeed used for solving the same search task. The overall results indicate that the tasks extracted by the proposed task-subtask extraction algorithm are indeed better than those extracted by the baselines.

For the second judgment study used for evaluating the quality of the hierarchy, we show 10 pairs of parent-child questions in each HIT and ask the human annotators to judge the subtask validity and usefulness. Overall, per method we evaluate 300 such judgments resulting in over 1200 judgments and used maximum voting criterion from among the 3 judges to decide the final label for each instance. Table 4 compares the performance of the proposed hierarchy extraction method against other hierarchical baselines. The identified subtask was found useful in 67% cases with the best performing baseline being useful in 52% of judged instances. This highlights that the extracted hierarchy is indeed composed of better subtasks which are found to be useful in completing the overall task depicted by the parent task. It is interesting to note that for BHCD and BAC baselines, most often the subtasks were found to be invalid and not useful.

Since the same parent-child task-subtask was judged for validity and usefulness, it is expected that the proportion of task-subtasks judged useful would be less than the ones judged valid. Indeed, as can be seen from the Table 4, the relative proportions of tasks-subtasks found useful is much less than those found valid.

## 5.3 Term Prediction

In addition to task extraction and user study based evaluation, we chose to follow an indirect evaluation approach based on *Query Term Prediction* wherein given an initial set of queries, we predict future query terms the user may issue later in the session.This is in line with our goal of supporting users tackling complex search tasks since a task identification system which is capable of identifying "*good*" search tasks will indeed perform better in predicting the set of future query terms.

To evaluate the performance of the proposed task extraction method, we primarily work with the TREC Session Track 2014 [? ] and AOL log data and constructed a new dataset consisting of user sessions from AOL logs concerned with Session Track queries. The session track data consists of over 1200 sessions while AOL logs consists of 20M search queries issued by over 657K users. We find the intersection of queries between the Session Track data and AOL logs to identify user sessions in AOL data trying to achieve similar task objectives. The Session Track dataset consists of 60 different ***topics***. For each of these 60 topics, we separately find user sessions from the entire AOL logs which contain query overlaps with these topics. For each topic, we iterate through the entire AOL logs and select any user session which contains query overlap with the current topic. As a result, we obtain a total of 14030 user sessions which contain around 6.4M queries.

Given the initial queries from a user session and a set of tasks extracted from Session Track data, we leverage queries from the identified task to predict future query terms. For each Session Track topic, we construct a task hierarchy and use the constructed task hierarchy to predict future query terms in the associated user sessions. More specifically, for each topic, we split each user session into two parts: (i) task matching and (ii) held-out evaluation part. We use queries from the task matching part of user sessions to obtain the right node in the task hierarchy from which we then recommend query terms. We pick the tree node which has the highest cosine similarity score based on all the query terms under consideration. We evaluate based on the absolute recall scores - the average number of recommended query terms which match with the query terms in the held-out evaluation part of user sessions.

We baseline against the top performing task extraction baselines from Section 5.1 as well as the top performing hierarchical algorithms from Section 5.2. To make fair comparisons, we consider nodes at the bottom most level of the pruned tree for task matching and term recommendation.

Figure 3 compares the performance on term prediction against the considered baselines. We plot the average number of query terms predicted against the proportion of user session data used. The proposed method is able to better predict future query terms than a standard task extraction baseline as well as a very recent hierarchy construction algorithm.

## 6 CONCLUSION

Search task hierarchies provide us with a more naturalistic view of considering complex tasks and representing the embedded task-subtask relationships. In this paper we first motivated the need for considering hierarchies of search tasks & subtasks and presented a novel bayesian nonparametric approach which extracts such hierarchies. We introduced a conjugate query affinity model to capture query affinities to help in task extraction. Finally, we propose the idea of Task Coherence and use it to identify atomic tasks. Our

experiments demonstrated the benefits of considering search task hierarchies. Importantly, we were able to demonstrate competitive performance while at the same time outputting a richer and more expressive model of search tasks. This expands the scope for better task recommendation, better search personalization and opens up new avenues for recommendations specifically targeting users based on the tasks they are involved in.

## REFERENCES


[1] Eugene Agichtein, Ryen W White, Susan T Dumais, and Paul N Bennet. 2012. Search, interrupted: understanding and predicting search task continuation. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 315–324.

[2] Ricardo Baeza-Yates, Carlos Hurtado, and Marcelo Mendoza. 2004. Query recommendation using query logs in search engines. In *International Conference on Extending Database Technology*. Springer, 588–596.

[3] Charles Blundell and Yee Whye Teh. 2013. Bayesian hierarchical community discovery. In *Advances in Neural Information Processing Systems*. 1601–1609.

[4] Charles Blundell, Yee Whye Teh, and Katherine A Heller. 2012. Bayesian rose trees. *arXiv preprint arXiv:1203.3468* (2012).

[5] Huanhuan Cao, Daxin Jiang, Jian Pei, Enhong Chen, and Hang Li. 2009. Towards context-aware search by learning a very large variable length hidden markov model from search logs. In *Proceedings of the 18th international conference on World wide web*. ACM, 191–200.

[6] Lara D Catledge and James E Pitkow. 1995. Characterizing browsing strategies in the World-Wide Web. *Computer Networks and ISDN systems* 27, 6 (1995), 1065–1073.

[7] Shui-Lung Chuang and Lee-Feng Chien. 2002. Towards automatic generation of query taxonomy: A hierarchical query clustering approach. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*. IEEE, 75–82.

[8] Debora Donato, Francesco Bonchi, Tom Chi, and Yoelle Maarek. 2010. Do you want to take notes?: identifying research missions in Yahoo! search pad. In *Proceedings of the 19th international conference on World wide web*. ACM, 321–330.

[9] Ahmed Hassan Awadallah, Ryen W White, Patrick Pantel, Susan T Dumais, and Yi-Min Wang. 2014. Supporting complex search tasks. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 829–838.

[10] Daqing He, Ayşe Göker, and David J Harper. 2002. Combining evidence for automatic web session identification. *Information Processing & Management* 38, 5 (2002), 727–742.

[11] Katherine A Heller and Zoubin Ghahramani. 2005. Bayesian hierarchical clustering. In *Proceedings of the 22nd international conference on Machine learning*. ACM, 297–304.

[12] Wen Hua, Yangqiu Song, Haixun Wang, and Xiaofang Zhou. 2013. Identifying users' topical tasks in web search. In *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM, 93–102.

[13] Rosie Jones and Kristina Lisa Klinkner. 2008. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 699–708.

[14] Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. 2006. Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web*. ACM, 387–396.

[15] Alexander Kotov, Paul N Bennett, Ryen W White, Susan T Dumais, and Jaime Teevan. 2011. Modeling and analysis of cross-session search tasks. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, 5–14.

[16] Dawn J Lawrie and W Bruce Croft. 2003. Generating hierarchical summaries for web searches. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. ACM, 457–458.

[17] Liangda Li, Hongbo Deng, Anlei Dong, Yi Chang, and Hongyuan Zha. 2014. Identifying and labeling search tasks via query-based hawkes processes. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 731–740.

[18] Zhen Liao, Yang Song, Li-wei He, and Yalou Huang. 2012. Evaluating the effectiveness of search task trails. In *Proceedings of the 21st international conference on World Wide Web*. ACM, 489–498.

[19] Xueqing Liu, Yangqiu Song, Shixia Liu, and Haixun Wang. 2012. Automatic taxonomy construction from keywords. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1433–1441.

[20] Claudio Lucchese, Salvatore Orlando, Raffaele Perego, Fabrizio Silvestri, and Gabriele Tolomei. 2011. Identifying task-based sessions in search engine query logs. In *Proceedings of the fourth ACM international conference on Web search and data mining*. ACM, 277–286.

[21] Claudio Lucchese, Salvatore Orlando, Raffaele Perego, Fabrizio Silvestri, and Gabriele Tolomei. 2013. Discovering tasks from search engine query logs. *ACM Transactions on Information Systems (TOIS)* 31, 3 (2013), 14.

[22] Bonnie Ma Kay and Carolyn Watters. 2008. Exploring multi-session web tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1187–1196.

[23] Rishabh Mehrotra, Prasanta Bhattacharya, and Emine Yilmaz. 2016. Characterizing users' multi-tasking behavior in web search. In *Proceedings of the 2016 ACM on Conference on Human Information Interaction and Retrieval*. ACM, 297–300.

[24] Rishabh Mehrotra and Emine Yilmaz. 2015. Terms, topics & tasks: Enhanced user modelling for better personalization. In *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*. ACM, 131–140.

[25] Rishabh Mehrotra and Emine Yilmaz. 2015. Towards hierarchies of search tasks & subtasks. In *Proceedings of the 24th International Conference on World Wide Web*. ACM, 73–74.

[26] Qiaozhu Mei, Hui Fang, and ChengXiang Zhai. 2007. A study of Poisson query generation model for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 319–326.

[27] Qiaozhu Mei, Dengyong Zhou, and Kenneth Church. 2008. Query suggestion using hitting time. In *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 469–478.

[28] Dan Morris, Meredith Ringel Morris, and Gina Venolia. 2008. SearchBar: a search-centric web history for task resumption and information re-finding. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1207–1216.

[29] David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic evaluation of topic coherence. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 100–108.

[30] Brendan O'Connor, Michel Krieger, and David Ahn. 2010. TweetMotif: Exploratory Search and Topic Summarization for Twitter.. In *ICWSM*. 384–385.

[31] Pavel Pecina. 2010. Lexical association measures and collocation extraction. *Language resources and evaluation* 44, 1-2 (2010), 137–158.

[32] Eran Segal and Daphne Koller. 2002. Probabilistic hierarchical clustering for biological data. In *Proceedings of the sixth annual international conference on Computational biology*. ACM, 273–280.

[33] Craig Silverstein, Hannes Marais, Monika Henzinger, and Michael Moricz. 1999. Analysis of a very large web search engine query log. In *ACm SIGIR Forum*, Vol. 33. ACM, 6–12.

[34] Adish Singla, Ryen White, and Jeff Huang. 2010. Studying trailfinding algorithms for enhanced web search. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. ACM, 443–450.

[35] Amanda Spink, Sherry Koshman, Minsoo Park, Chris Field, and Bernard J Jansen. 2005. Multitasking web search on vivisimo. com. In *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on*, Vol. 2. IEEE, 486–490.

[36] Hongning Wang, Yang Song, Ming-Wei Chang, Xiaodong He, Ahmed Hassan, and Ryen W White. 2014. Modeling action-level satisfaction for search task satisfaction prediction. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 123–132.

[37] Hongning Wang, Yang Song, Ming-Wei Chang, Xiaodong He, Ryen W White, and Wei Chu. 2013. Learning to extract cross-session search tasks. In *Proceedings of the 22nd international conference on World Wide Web*. ACM, 1353–1364.

[38] Ryen W White, Wei Chu, Ahmed Hassan, Xiaodong He, Yang Song, and Hongning Wang. 2013. Enhancing personalized search by mining and modeling task behavior. In *Proceedings of the 22nd international conference on World Wide Web*. ACM, 1411–1420.

[39] Hui Yang. 2012. Constructing task-specific taxonomies for document collection browsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, 1278–1289.

[40] Hui Yang. 2015. Browsing hierarchy construction by minimum evolution. *ACM Transactions on Information Systems (TOIS)* 33, 3 (2015), 13.

[41] Yongfeng Zhang, Min Zhang, Yiqun Liu, Chua Tat-Seng, Yi Zhang, and Shaoping Ma. 2015. Task-based recommendation on a web-scale. In *Big Data (Big Data), 2015 IEEE International Conference on*. IEEE, 827–836.