Leveraging Procedural Knowledge for Task-oriented Search

Zi Yang
Language Technologies Institute
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213
ziy@cs.cmu.edu

Eric Nyberg
Language Technologies Institute
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213
ehn@cs.cmu.edu

ABSTRACT

Many search engine users attempt to satisfy an information need by issuing multiple queries, with the expectation that each result will contribute some portion of the required information. Previous research has shown that structured or semi-structured descriptive knowledge bases (such as Wikipedia) can be used to improve search quality and experience for general or entity-centric queries. However, such resources do not have sufficient coverage of procedural knowledge, i.e. what actions should be performed and what factors should be considered to achieve some goal; such procedural knowledge is crucial when responding to task-oriented search queries. This paper provides a first attempt to bridge the gap between two evolving research areas: development of procedural knowledge bases (such as wikiHow) and task-oriented search. We investigate whether task-oriented search can benefit from existing procedural knowledge (*search task suggestion*) and whether automatic procedural knowledge construction can benefit from users' search activities (*automatic procedural knowledge base construction*). We propose to create a three-way parallel corpus of queries, query contexts, and task descriptions, and reduce both problems to sequence labeling tasks. We propose a set of textual features and structural features to identify key search phrases from task descriptions, and then adapt similar features to extract wikiHow-style procedural knowledge descriptions from search queries and relevant text snippets. We compare our proposed solution with baseline algorithms, commercial search engines, and the (manually-curated) wikiHow procedural knowledge; experimental results show an improvement of +0.28 to +0.41 in terms of Precision@8 and mean average precision (MAP).

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

Keywords

procedural knowledge base; search log; query suggestion; wikihow; search intent

SIGIR'15, August 09–13, 2015, Santiago, Chile.

ACM 978-1-4503-3621-5/15/08...$15.00
DOI: http://dx.doi.org/10.1145/2766462.2767744 .

1. INTRODUCTION

Many search engine users attempt to satisfy an information need by issuing multiple queries, with the expectation that each subtask will contribute some portion of the required information[1]. For example, to organize a conference (task), the organizer needs to look for information related to choosing a hotel, identifying and deciding among banquet options, recruiting volunteers, selecting and contacting publishers, etc (subtasks). For each specific task, users must gather more detailed information. For example, when choosing a hotel, the organizer must consider the number and size of conference rooms, potential arrangements for meal catering and menu planning, whether or not a discounted rate will be available, etc. There is a huge demand for search engines to better assist their users in achieving their intended goals for such *ad hoc* tasks.

Researchers have studied problems related to search intent analysis for general queries; for example, how to identify search intents [13, 11, 37, 25, 9, 19, 24, 28], how to suggest search queries to the user [3, 12, 32, 22, 41], how to rank results that cover diverse aspects of the task, etc. Most studies have so far mostly relied on queries and search logs, search result texts, behavioral information, etc. to improve search quality for entity-centric queries [15]. Recently, research has shown that structured knowledge bases (such as Wikidata or formerly FreeBase[2]) or semi-structured knowledge bases (such as Wikipedia[3]) can be used to improve search quality and experience [14, 6, 26, 43]. However, such knowledge bases focus on representing *descriptive knowledge* (also known as *declarative* or *propositional* knowledge), i.e. the knowledge of the attributes and features of things, and do not have sufficient coverage of *procedural knowledge* used for task-based problem solving [2, 18]. In the past decade, hundreds of thousands of how-to guides have been created by the community and/or professional editors on sites such as wikiHow[4] or eHow[5], which provide procedural knowledge in a semi-structured text format. There have also been attempts to define ontologies [7] to represent procedural knowledge, and algorithms have been proposed to automatically construct structured procedural knowledge bases such as PAA [1] from semi-structured procedural knowledge [34, 23, 1, 36]. However, to the best of our knowledge, structured or semi-structured procedural knowledge has not been studied in the context of task-oriented search as a means to improve search quality and experience.

[1] *(Sub)task* in this paper refers to an action that is performed to achieve a goal in a general context [29], which differs from the notion of *task* in the search scenario [20, 21]. We use *(sub) search task* to denote the latter case.
[2] http://wikidata.org/
[3] http://wikipedia.org/
[4] http://wikihow.com/
[5] http://ehow.com/

This paper provides a first attempt to bridge the gap between the two evolving research areas: procedural knowledge base and task-oriented search. We investigate whether task-oriented search can benefit from an existing procedural knowledge base and whether automatic procedural knowledge construction can benefit from users' search activities.

- In particular, we first study how to define features that capture the procedural knowledge writing style, in order to extract queryable phrases from descriptions of the tasks related to the user's information need. We use these features to design a supervised framework and to automatically build a three-way parallel corpus without manual annotation.
- We also study how to construct a procedural knowledge base from users' raw search data (e.g. search logs) as well as data aggregated from multiple users (e.g. suggested queries), based on the hypothesis that the overall goal of task-oriented search is accomplished through a series of searches using multiple queries, which are either reformulated by users or suggested by search engines and accepted by users.

We attempt to leverage the style guidelines for writing semi-structured procedural knowledge, such as the wikiHow guide[6], which indicates that an action-oriented instruction beginning with a verb is required at the beginning of each procedural step, but we also attempt to process articles that do not fully comply with the guide. We accordingly propose a set of textual features and structural features to identify query spans from each task description, and then adapt similar features to extract wikiHow-style procedural knowledge descriptions from search queries and relevant text snippets. We compare our proposed solution with baseline algorithms as well as commercial search engines and the manually-curated procedural knowledge base wikiHow for both problems; experimental results show an improvement of +0.28 to +0.41 in terms of Precision@8 and mean average precision (MAP). Our work can directly benefit search task mining problems such as the NTCIR-11 IMine TaskMine task[7] [29]; automatically extracted procedural knowledge can also provide decision processes for retrieval-based decision support systems such as QUADS [42].

2. RELATED WORK

In this section, we review related work in search intent analysis and task-oriented search, for which we summarize applications focusing on query suggestion. We then survey prior work on procedural knowledge acquisition and its application, focusing on ontology definition for procedural knowledge and algorithms to automatically extract semi-structured procedural knowledge from text.

2.1 Search intent & task-oriented search

Broder [8] and Rose and Levinson [38] propose to categorize search queries into *informational*, *navigational*, and *transactional* and relate them to a more fine-grained *goal hierarchy*. Marchionini [30] analyzed the user's high-level "need to know" and introduced the concept of *exploratory search* as an extension of the traditional *lookup*. Researchers have also proposed to predict both high-level and specific search intents from textual and behavioral information such as Web page contents [13], search queries [11], click logs [37, 25, 9], mouse movements and scrolling [19], and results of individual user questionnaires [24, 28]. Identifying searchers' intents and incorporating them into the search process can contribute to search query suggestion and prediction [3, 12, 32, 22, 41] and selection of exact responses (such as various Google search special features) [40]. The suggested query type depends on the original search query and the predicted intent. E.g., when a user searches for an entity, search engines may suggest other relevant entities [6, 5], actions paired with the entity [27], or attributes [14], mostly using declarative knowledge bases such as Freebase or Wikipedia. However, whether a task-oriented search can achieve an analogous benefit from a procedural knowledge base has not been studied.

In the area of search intent analysis, Hassan et al. [20, 21] studied the *complex search task*, a search behavior that is usually applied to task-oriented search, using search queries. In comparison, our work focuses specifically on task-oriented search, and ignores other types of search (such as browsing different attributes of an object), which allows us to take the advantage of existing procedural knowledge to more reliably support search tasks (when compared to the use of general search logs). We also study to how to build a procedural knowledge base with a full text representation of task-oriented knowledge. The work by Weber et al. [40] on extracting tips from Yahoo! Answers to address how-to Web queries also incorporates task-oriented question and tip extraction from unstructured question-answer texts, whereas our work deals with only complex task-related queries and focuses on identifying queryable snippets from more structured procedural knowledge. Moreover, we also study how to construct procedural knowledge bases from search queries and result snippets, which can be extended to facilitate automatic question answering for community-based QA.

More recently, the IMine Subtopic Mining subtask [29] and the pilot subtask TaskMine at NTCIR focuses on identifying subtasks for a given complex task, and is highly relevant to our work. For a given task (e.g. "lose weight"), possible outputs are expected to be subtask steps (e.g., "do physical exercise", "measure calorie intake"). We make a similar assumption that the given queries are focused on seeking information (subtasks) for complex tasks.

2.2 Procedural knowledge acquisition

Cognitive psychology defines "procedural knowledge" (or imperative knowledge) as the knowledge exercised in the performance of some task [2, 18]. The Semantic Web community has attempted to formally define ontologies to represent procedural knowledge [7], which usually include an *instruction* (or *action* sequence) and a *purpose* (or *goal*). Also defined in such ontologies are the relations between procedural knowledge elements, at different levels of granularity, such as *has-step*, *has-goal* [34] or *is-achieved-by* [17].

Several automatic procedural knowledge base construction approaches have also been proposed to extract instructions from semi-structured texts, e.g. eHow or wikiHow articles [34, 23, 1, 36], recipes [36], community-based QA [4], etc. Most approaches take advantage of structural information (e.g. HTML tags [4], enumeration indicators [16, 23]), and define rules or templates to extract textual content. In a separate step, NLP tools are applied to extract relations and normalize each goal and action to its ontological form, to support linking to other ontological resources. Researchers have also studied how to identify script knowledge [39] which focuses on the temporal ordering of events [10].

In contrast, our approach takes advantage of the writing style of semi-structured procedural texts and proposes a set of structural and textual features for identifying procedural knowledge; the implemented approach can be optimized with a supervised learning method. Moreover, beyond the conventional use of small-scale procedural knowledge in AI planning [33] or NLP [4], this paper studies the problem of how to apply an online large-scale procedural knowledge base to complex task search problems.

[6] http://wikihow.com/wikiHow:Writer's-Guide

[7] http://dl.kuis.kyoto-u.ac.jp/ntcir-11/taskmine/

3. PROBLEM DEFINITION

In this section, we first give our formal definition of *procedural knowledge base*, and then introduce the two main problems we will study in this paper: *search task suggestion using procedural knowledge base (STS)* and *automatic procedural knowledge base construction from search activities (APKBC)*.

Many knowledge bases such as Wikipedia or Wikidata that have been widely utilized contain a huge amount of *descriptive knowledge*. Procedural knowledge [2, 18], also known as *know-how*, is the knowledge exercised in the accomplishment of a task, i.e. how to do things, which is usually acquired by experience and considered *tacit* and not easily shared, compared to *descriptive knowledge*. However, shared explicit procedural knowledge lays a foundation for efficiently coordinated action, which is often referred to as *best practices* or *business rules* within communities or organizations. As wikiHow and many similar how-to Web sites allow users to easily share procedural knowledge, semi-structured large-scale procedural knowledge bases have become available. People have also attempted to construct structured procedural knowledge bases by defining ontologies or incorporating other existing representations such as PDDL[8] [34, 23, 1, 36]. Since we notice that many of the proposed definitions use different terminologies, e.g. *goal* vs. *target* vs. *purpose*, *instruction* vs. *action sequence*, *step* vs. *action*, etc, although their meanings are identical, we first formally define the terminology related to procedural knowledge base.

Definition 1 (Task)**.** *A **task** t is represented by a short and concise **summary** and a detailed **explanation** for clarification.*

Definition 2 (Is-achieved-by relation)**.** *A task is connected with an ordered list of n tasks $s_1, \ldots, s_n$ (referred to as **subtasks**) that contribute to the task t by **is-achieved-by** relations $r(t, s)$. A* weight *$w(t, s)$ may also associate with each relation $r(t, s)$. We assume the weights associated with the outgoing relations should sum to one, i.e. $\sum_i r(t, s_i) = 1$.*

Definition 3 (Procedural knowledge graph)**.** ***Procedural knowledge graph** $G = (T, R)$ is the set of all tasks T and all is-achieved-by relations R.*

Definition 4 (Plan, goal, and primitive tasks)**.** *A **plan** is a subgraph of G, which is a tree with root being the **goal** and the leaves being the **primitive tasks**.*

A *semi-structured procedural knowledge base* such as wikiHow has an identical structure, but it allows users to more easily view and edit; each article corresponds to a task, whose summary is given in the title and whose explanation is detailed in the introduction section. Each step (or substep) also corresponds to a task. The is-achieved-by relations have three representation forms: numbered *steps* in each article page (first-level subtasks), bulleted substeps under each step description (second-level subtasks), and free links[9] directed from a step to another task page. Steps that have no substeps or outgoing free links are considered *primitive tasks*. Current semi-structured procedural knowledge bases do not allow editors to explicitly specify the the relation strength between tasks, or the importance of each subtask[10]. Therefore, we assume each relation shares an equal weight with all other outgoing relations, i.e. $w(t, s) = 1/\text{od}(t)$, where od is the out-degree of the task t.

Procedural knowledge captures how people decompose a complex task into a mixture of several mental actions (e.g. "choose", "determine", etc.) and/or physical actions (e.g. "visit", "store", etc.). When users turn to search engines for information seeking and problem solving, we investigate how existing procedural knowledge can be leveraged to help understand a user's search intent and suggest sub search tasks to users.

Problem 1 (**Search task suggestion using procedural knowledge, STS**)**.** *Given a procedural knowledge graph G and a task-oriented search task q, we aim to (a) identify the task $t \in T$ the user intends to accomplish, and then (b) retrieve a list of k subtasks $s_1, \ldots, s_k \in T$ and also (c) suggest the corresponding sub search tasks $p_1, \ldots, p_k$.*

A search task is usually represented by a search query or a key phrase [29, 20, 21]. Similarly, we assume both the input search task q and the suggested search tasks $p_1, \ldots, p_k$ are queryable phrases. As a result, the problem can be straightforwardly solved by suggesting relevant (sub)queries for task-oriented queries. In contrast to the entities and attributes in a descriptive knowledge base, which are usually words or phrases and can used directly in query suggestion, summaries and explanations in a procedural knowledge usually contain richer action and condition information. Therefore, the problem poses three challenges. First, we need to identify the task $t \in T$ the user intends to accomplish. If the query corresponding to the search task exactly matches the summary of a task t (exact string match or after lexical transformations), then we can directly return t. However, in most cases when an exact match does not exist, we need to incorporate additional information such as the search result page of the query and the full description of each task, and rank candidate tasks using a retrieval method. Second, we need to choose the tasks $s_1, \ldots, s_k \in T$ to suggest and then prioritize them based on the knowledge G containing relations V. Intuitively, we can choose the task t's top-k subtasks ordered by the weight of the is-achieved-by relation. Finally, we need to extract queryable phrases from each task description, and the algorithm should thus learn how searchers have been trained to formulate queries.

Although community-centric procedural knowledge bases may have collected hundreds of thousands of task descriptions, users still face *ad hoc* situations (tasks) that are not covered by an existing procedural knowledge base. However, we hypothesize that other searchers may have faced similar situations in the past and have already interacted with search engines to attempt a solution, which means we may discover implicitly expressed procedural knowledge from users' raw search activities (e.g. search logs) as well as those aggregated from many searchers (e.g. suggested queries). Therefore, we investigate whether we can automatically construct a procedural knowledge base like wikiHow using search queries and relevant documents returned from search engines.

Problem 2 (**Automatic procedural knowledge base construction from search activities, APKBC**)**.** *Given a task t, we aim to (a) identify a search task q, and (b) collect k related search tasks $p_1, \ldots, p_k$, and then (c) identify n ($\leq k$) search tasks $p_{i_1}, \ldots, p_{i_n}$ to generate n tasks $s_1, \ldots, s_n$ that can be performed to accomplish the task t with text description.*

Related search tasks can be identified from users' search activities [37, 25, 9] and/or suggested by search engines [3, 12, 32, 22, 41]. Depending on the search intent, related actions, attributes, entities, or any form of reformulation of the original search task may be considered related, which however do not necessarily embed

[8] http://www.plg.inf.uc3m.es/ipc2011-deterministic/attachments/Resources/kovacs-pddl-3.1-2011.pdf

[9] http://en.wikipedia.org/wiki/Help:Wiki_markup#Free_links

[10] As users may verbally express their confidence and/or their attitude toward the importance in the task explanation section, we can hence incorporate richer linguistic features such as modality particles, verbal auxiliaries, etc. to identify verbal expression of weight.

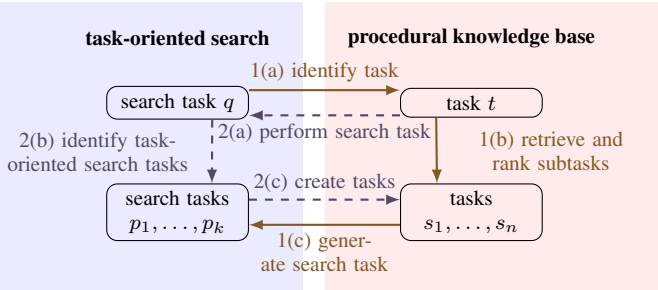

Figure 1: Subproblems in *search task suggestion using procedural knowledge* problem (brown solid arrows and brown labels) and *automatic procedural knowledge base construction from search activities* problem (purple dashed arrows and purple labels)

procedural knowledge. Therefore, the first challenge in extracting procedural knowledge is to identify the is-achieved-by relations, i.e., which of them correspond to the tasks that can achieve the goal. A textual description is a less ambiguous representation of a task in a procedural knowledge base, which can be accessed both by humans and by automatic decision support systems that require predefined natural language decision processes as input (e.g. QUADS [42]). Our second challenge is therefore to automatically generate text descriptions for procedural knowledge which attempt to conform to a common writing style. We summarize the interactions between task-oriented search and procedural knowledge base construction in Figure 1.

4. PROPOSED APPROACH

In this section, we propose a supervised learning framework, which requires zero extra manual annotation, and instead takes advantage of available artifacts; users' search activities and manually curated procedural knowledge are used to train models for both STS and APKBC problems. In particular, existing search logs can reveal how searchers have been trained to formulate queries, but they don't necessarily or sufficiently cover how to search for procedural knowledge. On the other hand, existing procedural knowledge bases indicate how to accomplish tasks in a very comprehensive manner, but are not necessarily optimized for interactive search. We first describe how to build a *three-way parallel corpus* for training, and then define linguistic and structural features for training sequence labeling models for both problems. Then, we present detailed procedures for suggesting search tasks using procedural knowledge and constructing procedural knowledge bases from search logs.

4.1 Three-way parallel corpus construction

We create a three-way parallel corpus that contains triples $\{\langle q, t, c\rangle\}$ with a query q, a matching task t, and a context c of q and t, which is a passage of text that describes the task. We build the three-way parallel corpus in an iterative manner.

We first identify the exact matching pairs of searchers' issued search tasks and task summaries in the procedural knowledge base, which guarantees precision in the first step, although we expect that many lexically distinct but semantically identical pairs may exist that will not be extracted. To achieve this, we may scan through the entire search query log to find each query q that matches the description of the task t. We then retrieve the context c by extracting the textual content from the top documents relevant to the description of the task t. We add the triple $\langle q, t, c\rangle$ to the parallel corpus. In addition, we collect related queries by combining the user-issued queries from the same session and the list of queries suggested by the search engine for the next iteration.

If we have no large query log, we may also manually create task-oriented search tasks for the tasks in the procedural knowledge base. Specifically, we use the summary of the task t to form a search query q and issue it to the search engine to extract the context c. Since these search queries are "artificially" created, and do not necessarily represent how searchers normally formulate queries for these tasks in similar situations, we exclude the triple $\langle q, t, c\rangle$ that contains the original query q from the parallel corpus, which is only used for search engines to suggest real queries for the next iteration.

Once we collect a list of related queries for q, we compare them with the known subtasks of t, i.e. the tasks connected by is-achieved-by relations in the procedural knowledge base. For each related query p, we first find the subtasks $s_1, \ldots, s_n$ that contain p in either its summary or explanation, and retrieve its context d. We then add all $\{\langle p, s_i, d\rangle\}_{i\in[1,n]}$ to the parallel corpus. As we assume that all necessary subtasks have been detailed in the procedural knowledge base, we consider a related query irrelevant to accomplishing the task if it is not mentioned in any of the subtasks, and it is excluded in the parallel corpus and discarded for future iterations, which solves Problem 2(b). We also ignore subtasks that are not matched to any related query, a situation that may arise from a lack of evidence from query logs, or truncation due to limited space for displaying suggested queries on a result page. We continue the iterative process.

Furthermore, we annotate the occurrence of the query q in the task t's description. We also similarly annotate the occurrence of the summary and explanation of the task t in the context c by finding the contiguous sequence of words (a passage) from the context that is most relevant to the task summary or explanation. An example is shown in Figure 2. For training a sequence labeling model, we employ the `BIO` encoding, i.e., each token in the task description (or context) that begins each occurrence of the query phrase (or the task summary or explanation) is labeled as $\texttt{B}^{\texttt{Q}}$ (or $\texttt{B}^{\texttt{TS}}$, $\texttt{B}^{\texttt{TE}}$), the tokens that continue in the query phrase (or the task summary or explanation) are labeled as $\texttt{I}^{\texttt{Q}}$ (or $\texttt{I}^{\texttt{TS}}$, $\texttt{I}^{\texttt{TE}}$), and all other tokens in the task (or the context) are labeled as `O`.

4.2 Feature definition and model construction

We use the constructed parallel corpus, which contains triples of queries, task representations, and contexts, to train supervised models that can automatically create task-oriented queries for tasks, as well as task descriptions for task-oriented search tasks. We view both problems as a word sequence extraction problem, which is usually reduced to a sequence labeling task. Therefore, we define the features by taking advantage of the writing style prevalent in semi-structured procedural knowledge bases, and then apply a common statistical model for sequence labeling (such as Conditional Random Fields (CRF)), which also enables us to process articles that do not fully comply with a style guide.

We define both syntactic features and structural (positional) features that are motivated by the writing guide as follows:

Location (LOC). As suggested in the wikiHow writing guide, a task should provide both "skimmable information that readers can quickly understand" in the title of the article and in the beginning sentence of each step, and "rich informative articles for more patient users" in the article's introduction and in the sentences which follow in the detailed explanation of each step. Therefore, we define features to capture the location of each word.

Part of speech (POS). Since both the article title and the first sentence in each step are required to begin with a verb in bare in-

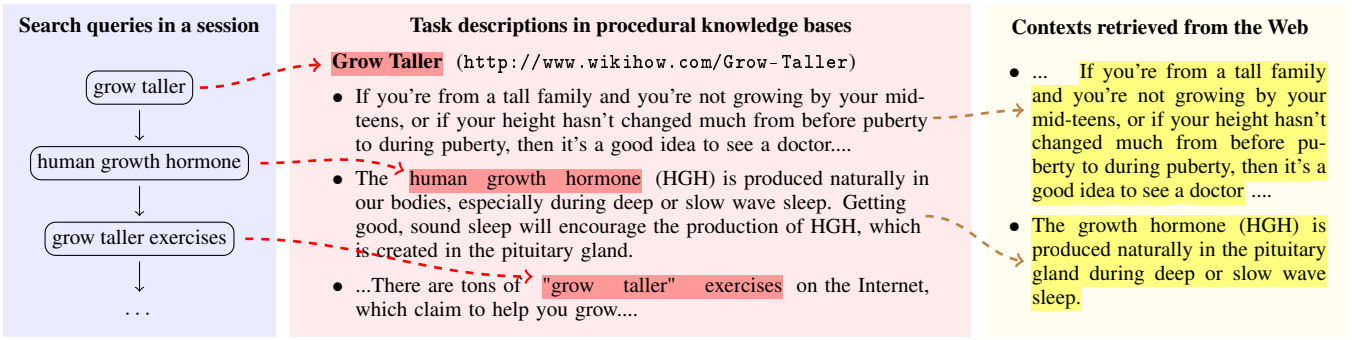

Figure 2: An example of the parallel corpus construction process

finitive form, we also extract the fine-grained part-of-speech value for each token.

Parsing (PAR). To further understand the *task facets* such as occurrences of subsidiary resources (e.g. a target object) or constraints (e.g. a duration), we include features extracted from dependency parsing, named entity extraction, and chunking.

Word and context. As in other natural language sequence labeling tasks, we also define features including the token's surface form, its stem form, its TF-IDF score, as well as the word features and the part-of-speech tags of the previous and the next words in the task summary or explanation.

We extract the same set of features (except the location feature) from both the context c (X^c) and task description t (X^t) for each triple in the parallel corpus, which together with the corresponding annotated BIO label (Y^c and Y^t) comprises a training instance for sequence labeling. We train a query construction model $M^{\mathtt{Q}}$ using the training sets (X^t, Y^t), a task summary construction model $M^{\mathtt{TS}}$ and a task explanation construction model $M^{\mathtt{TE}}$ using (X^c, Y^c).

To construct search queries for an unseen task t, we first extract the features x^t from the task description, and then apply the query construction model $M^{\mathtt{Q}}$ to extract each search query candidate q_i, i.e. to identify each word sequence $\mathbf{w}_i = w_{i_1} \dots w_{i_{l_i}}$ from t with the corresponding $\mathbf{y}_i^{t*}$ labeled as $\mathtt{B^Q I^Q} \dots \mathtt{I^Q}$, where l_i represents the length of $\mathbf{w}_i$, and $\mathbf{y}^{t*}$ is the optimal label sequence, i.e.,

$$\mathbf{y}^{t*} = \underset{\mathbf{y}^t \in \{\mathtt{B^Q,I^Q,O}\}^{|t|}}{\operatorname{argmax}} p(\mathbf{y}^t | \mathbf{x}^t; M^{\mathtt{Q}}) \quad (1)$$

where $|t|$ represents the length of the task t's description. Similarly, to construct task descriptions for an unseen query q, we first retrieve the context c and extract the features x^c from the context, and then apply the task description construction model M^c to extract each word sequence labeled as $\mathtt{B^{TS} I^{TS}} \dots \mathtt{I^{TS}}$ and $\mathtt{B^{TE} I^{TE}} \dots \mathtt{I^{TE}}$ as the summary and the explanation of a task t_i respectively from optimizing the following equation:

$$\mathbf{y}^{c*} = \underset{\mathbf{y}^c \in \{\mathtt{B^{T\cdot}, I^{T\cdot}, O}\}^{|c|}}{\operatorname{argmax}} p(\mathbf{y}^c | \mathbf{x}^c; M^{\mathtt{T\cdot}}) \quad (2)$$

4.3 Search task suggestion

Given a task-oriented search task represented by query q, we first retrieve a list of candidate tasks from the procedural knowledge base that mention the query q in either the summary or the explanation. When more than one task is returned from the procedural knowledge base, we need to determine which task is the best fit for the user's search intent. Therefore, we leverage the query construction model $M^{\mathtt{Q}}$ to estimate the likelihood of the j-th occurrence of the query q in a retrieved task t_i's description (i.e. a word sequence $\mathbf{w}_{ij} = w_{ij_1} \dots w_{ij_l}$, where l is the length of query q). We select the i^*-th task that contains the j^*-th word sequence which maximizes the conditional probability, i.e.

$$i^*, j^* = \underset{i,j}{\operatorname{argmax}}\, p(\mathbf{y}_{ij}^{t_i} = \mathtt{B^Q I^Q} \dots \mathtt{I^Q} | \mathbf{x}^{t_i}; M^{\mathtt{Q}}) \quad (3)$$

Once we have identified the task t, we identify the tasks in the procedural knowledge base for future query suggestion using a few heuristic strategies. First, we may select only the first-level subtasks and order the weight of the is-achieved-by relation in descending order. We may also select both the first-level and second-level subtasks and use the weight of each first-level subtask or the weight of each second-level subtask multiplied by the weight of its corresponding first-level subtask.

When a list of subtask candidates $s_1, \dots, s_n$ are retrieved from the existing procedural knowledge base, we apply the query construction model $M^{\mathtt{Q}}$ again, for each subtask s_i's summary and explanation to identify each word sequence $\mathbf{w}_{ij}$ labeled as $\mathtt{B^Q I^Q} \dots \mathtt{I^Q}$ using Eq. 1. Among the extracted query candidates $\{\mathbf{w}_{ij}\}_j$ for each subtask s_i, we choose the query p_i that maximizes the likelihood $p(\mathbf{y}_{ij}^{s_i} = \mathtt{B^Q I^Q} \dots \mathtt{I^Q} | \mathbf{x}^{s_i}; M^t)$. The "winning" query candidates $p_1, \dots, p_n$ corresponding to the subtask candidates are globally ranked by multiplying the weight obtained from subtask retrieval with the likelihood estimation; finally, we select the top-k queries. This process guarantees that the queries are extracted from distinct subtask candidates, which can lead to the most coverage of all necessary subtasks. Alternatively, a diversity-aware ranking approach can also be applied to global rank all query candidates $\{\mathbf{w}_{ij}\}_j$. A *plan* can be gradually implemented and detailed when we iteratively apply this approach.

4.4 Automatic procedural knowledge base construction

Given a task description t, we directly apply the query construction model $M^{\mathtt{Q}}$ to extract a task-oriented search query q using Eq. 1, and then (similar to the process in building the parallel corpus) we identify the the queries related to q in both search logs and suggested queries. The entire search session (reformulation of queries, click activities, etc.) can be analyzed to determine how a specific user works to accomplish a task; therefore, by mining the search session data, we should be able to model how users typically accomplish tasks. As a result, the search engine should be able to correctly suggest to the user related tasks, rather than related entities or attributes. Although this assumption may not always hold in real-world search scenarios, it allows us to consider how related tasks can be further explored to extract subtasks for t, as long as the task description construction model can identify any passage that can be used for the task summary or explanation. In Section 5, we observe whether or to what extent this assumption holds for actual search scenarios.

For each related query p_i, we collect its context by extracting relevant document snippets from search engines, and apply the task description construction model $M^{\mathrm{T}\cdot}$ to extract the most likely summary candidate $\mathbf{w_i}$ and explanation candidate $\mathbf{v_i}$ for task s_i, according to Eq. 2, among all extracted summary candidates and explanation candidates for s_i. We select the top-n tasks ordered by the estimated likelihood for their summaries.

5. EXPERIMENT

In this section, we describe the experiments conducted to evaluate the effectiveness of the approach proposed in Section 4. We first introduce the data sets for the experiments, and then we describe the experiment settings. The experimental results are presented in three subsections for the constructed three-way parallel corpus and the two problems we introduced in Section 3; we also analyze the limitations of the current query suggestion approaches for task-oriented search, and the coverage and quality of current procedural knowledge bases for search suggestion. The data and code used in this section are available to download[11].

5.1 Data Preparation

We used two publicly available data sets (an English wikiHow data dump and the AOL search log) and public search engines (Google and Bing) to collect the suggested queries and contexts.

English wikiHow data dump. We crawled an English wikiHow data dump using a modified version of the WikiTeam tool[12], which contains 198,163 non-redirect articles within namespace "0" (Main)[13]. We also filtered out the articles that are marked as `stub` (incomplete and need more information) or have no "Introduction" or "Steps", which results in a collection of 149,975 valid articles. We performed minimal pre-processing to reformat the articles to the MediaWiki format[14], without changing any textual content.

In particular, we first extracted the top-level task summary and explanation after identifying the title and the introduction of each article. We then located the "Steps" section and extracted the enumerated or bulleted items to build a local subtask hierarchy. Next, we built a procedural knowledge graph by creating nodes representing all the top-level tasks and their subtasks and establishing relations based on both the task-subtask relation as well as internal links. Applying this approach, the constructed procedural knowledge graph contains a total of 1,488,587 tasks and 1,439,217 relations, where 265,509 of the tasks are non-primitive, and 100,605 relations come from internal links. We built a Lucene[15] index for all task descriptions, which supports retrieval of candidate tasks for the search task suggestion problem described in Section 4.3.

AOL search query log. We also used the publicly available AOL search query log, which consists of over 21M Web queries (over 10M unique queries) collected over three months in 2006 from AOL [35]. We downcased the query strings and task summaries, and removed all non-alphanumeric characters. We identified that 867 unique queries (corresponding to 9,847 new queries from 23,099 lines) match some task summary in our constructed procedural knowledge base. In the experiment, we identified 639 unique queries that have at least two tokens (corresponding to 3,086 new queries from 7,019 lines), which tend to be less ambiguous and more likely task-oriented. We could estimate that each task-oriented search task is repeated 4.8 times on average, compared to 2.1 times for all queries, supporting the intuition that common task-oriented searches tasks are more often encountered than general search tasks, providing further motivation for our study. The 639 unique queries correspond to 619 tasks if punctuation marks and whitespaces are ignored.

To retrieve the related queries issued by users in the same session from the query log, we first identified related query candidates by collecting the queries that were issued by the same user within 30 minutes after they issued each matching query. In this way, we collected 33,548 query candidates (31,955 unique queries, 50 per query). We did not use other commonly adopted session detection heuristics such as edit distance or cosine similarity between query pairs, since a task-oriented search query may not share words in common with related search queries. For example, we identified a case where "visit niagara falls"[16] is followed by another query "map of northeast" in the AOL search log, "design a living room"[17] is followed by another query "choosing colors", etc. We instead counted on the three-way parallel corpus construction process to correctly identify which of the query candidates are relevant to accomplishing the task.

Queries suggested by search engines. To create "artificial" search tasks, we randomly sampled 1,000 non-primitive tasks from the constructed procedural knowledge that do not appear in the query log. We limited the total number of tasks in building the parallel corpus partly due to our limited access to commercial search engines, and also because in a preliminary experiment we found that performance of the sequence labeling model on the test set had become stable. We merged them with the 639 identified queries from AOL search query log, using commercial search engines (Google and Bing) to build the parallel corpus, which correspond to 1,619 tasks. We constructed each query without using additional operators such as quotes, plus signs, minus sign, etc., unless quotes appear in the summary. Google may suggest up to eight queries at the bottom of the search page for each search task and Bing may suggest up to sixteen queries within the search result column and on the right side of the results page. We collected a total of 9,906 queries suggested by Google (6.11 per query on average, 8 maximum) and 9,715 suggested queries by Bing (5.99 per query on average, 13 maximum).

We notice that both Google and Bing are able to display procedures directly on the result page from wikiHow (only Bing) or other procedural data sources (Google), which allows users to grasp the knowledge without an extra click, but hardly helps users or automatic decision support systems identify how to explore more related information from the Web. We also found that both commercial search engines tend to suggest and display related queries for short queries, as they are more likely topically broad and semantically ambiguous. However, for task-oriented search, the complexity of describing the task does not necessarily reflect the difficulty of accomplishing the task, in terms of which aspects to consider and the steps to perform. In Section 5.4, we further evaluate whether commercial search engines are effective for task suggestion.

Context extracted from search engines. We used Goolge to collect the contexts of the queries both in the parallel corpus (used for model training and testing) and those that are potentially useful for new procedural knowledge discard auxiliary words such as pronouns, articles, etc. when they formulate queries, retaining only nouns, verbs, adjectives, etc., we also plan to study whether we can

[11] `http://github.com/ziy/pkb`

[12] `http://github.com/WikiTeam/wikiteam/`. Unlike most MediaWiki sites, each article on wikiHow has a URL which is formed by concatenating the words in its title using a hyphen ("-") rather than an underscore ("_"), so we modified the code accordingly.

[13] `http://www.mediawiki.org/wiki/Manual:Namespace`

[14] `http://www.mediawiki.org/wiki/Help:Formatting`

[15] `http://lucene.apache.org/`, Ver. 4.10.1

[16] Also on wikiHow: `http://www.wikihow.com/Visit-Niagara-Falls`

[17] Also on wikiHow: `http://www.wikihow.com/Design-a-Living-Room`

extract queryable key words from non-continuous text spans.overy (used for end-to-end APKBC evaluation). We first extracted URLs for context candidates from their first search result pages, and excluded Web documents from the `wikihow.com` domain to increase the reliability and generalizability of the trained model, and also excluded `google.com` domains and URLs that have only domain names without subpaths, which are usually navigational search results. Then, we used the Boilerpipe[18] to extract textual content from the HTML documents. For queries in the parallel corpus, we finally downloaded 7,440 context documents and successfully processed 7,437 documents using Boilerpipe, and for end-to-end evaluation, we downloaded and extracted 3,512 context documents.

5.2 Experiment Settings

We describe the experiment settings, including the tools we used to extract features and learn the sequence labeling models, the evaluation methods, and baseline methods.

Evaluation methods. We conducted two types of evaluation for both problems. We first evaluate the performance of proposed sequence labeling approach on the query construction and task description construction tasks using the automatically labeled parallel corpus. Then, based on manual judgment, we evaluate our proposed end-to-end solution for both STS and APKBC problems, and compare with commercial search engine suggestions and the current wikiHow knowledge base, respectively.

We performed 10-fold cross validation on the constructed parallel corpus, and report average performance results. We extracted text spans from the predicted labels and used precision, recall and F-1, averaged over the performance on all test instances (macro-averaged) and averaged on each task then across all tasksalso (micro-averaged), to compare the proposed approach with baseline methods. We also employed two ROUGE scores (F-1 based ROUGE-2 and ROUGE-S4[19]), after we removed common English stop words[20] and constructed an array of words in lemma forms.

In the end-to-end comparison, we randomly sampled 50 triples from the parallel corpus and we applied both our approach for search task suggestion and commercial search engine services to produce up to eight related queries. We manually judged whether each search query can obtain a search result that can help achieve the user's goal. Similarly, we applied our procedural knowledge base construction approach to generate up to eight procedural descriptions and merged it with the wikiHow "steps". We manually judged whether each subtask summary and explanation that the system produced can be considered a valid "step" description for wikiHow. Given that this judgment is binary, we report the macro-averaged and micro-averaged Precision@8, and MAP averaged over all 50 test instances.

Baseline methods. For the query construction and task description construction tasks, we compare the proposed approach **CRF** with other supervised and unsupervised approaches. We varied the classifier to evaluate **HMM**, a Hidden Markov Model based sequence labeling approach based on surface forms, **LR**, a logistic regression classifier, and an **SVM** classifier. The latter two are trained on the same set of features to classify `B`, `I`, and `O` without the sequence labeling assumption. We also compared performance with models that use only one category of the proposed features (**W/**), all but one category (**W/O**), and only local or context features. Finally, we compare with a key word extraction method based on TF-IDF (**TFIDF**), where we try to maximize the macro-averaged F-1 score by tuning the TF-IDF score threshold when determining whether a word is selected as a key word.

Feature extractors and learners. We used Stanford CoreNLP[21] to extract sentences, tokens, stems, POS tags, dependency labels, chunks (noun phrases and verb phrases), and named entities using the default configuration. We used MALLET[22] [31] to learn and test the sequence labeling models (CRF and HMM) with `sparse` weighting method, LibLinear[23] for training and testing LR and SVM models. As the Web documents tend to be noisy, we preprocessed the texts by inserting periods at certain places to improve the parsing performance. Details can be found in the source code.

Table 1: Averaged number and percentage of related queries suggested by Google or Bing or issued by the users subsequently in the same session that are mentioned by a task description in wikiHow

	Averaged number			Percentage (%)		
	Google	Bing	Log	Google	Bing	Log
Full phrase	0.070	0.046	0.050	1.162	0.727	1.131
New words	0.473	0.592	0.164	8.487	9.522	4.639

5.3 Parallel corpus construction result

Including the 639 queries that have already been associated with some tasks in the procedural knowledge base, we identified that a total of 1,182 query-task description pairs (from 1,146 distinct descriptions) among the suggested queries can be found in the subtask descriptions of the task corresponding to the original query. We annotated the context with each corresponding task summary and explanation. Specifically, we segmented the context into sentences and selected those that contain all the tokens in the task summary or at least 70% of the tokens in the task explanation as task description embedding sentences, and annotated the minimal span that contains all overlapping tokens.

To further understand how many of the related queries are aligned to a task description, we show in Table 1 the averaged number and percentage of related queries suggested by Google or Bing or issued by users subsequently in the same session that are mentioned within a task description. "New words" refer to the subset of words that are in the suggested queries but not in the original queries. First, we can see that the queries identified in the same session do not have a higher quality (1.131% – 4.639%) in related task suggestion, which may be due to an over-simplified session detection algorithm. In comparison, related queries collected from commercial search engines tend to be more relevant (0.727% – 9.522%). Moreover, if we focus on the new words, we identify many more (from 0.727% – 1.162% to 8.487% – 9.522%) tasks that are aligned with the queries suggested by the search engines.

We note that if a query cannot be aligned with a task description using our proposed construction method, this does not guarantee that the query is not semantically identical (e.g. a paraphrase) of any task description. In fact, as motivated in Section 4.1, the surface form matching approach can simplify the parallel corpus construction process while still guarantee the parallelity.

5.4 Search task suggestion result

We first compare the query construction result between the proposed approach and the baseline methods in Table 2. We also con-

[18] `http://code.google.com/p/boilerpipe/`, Ver. 1.2.0

[19] We did not use any unigram based metric (such as ROUGE-1 or ROUGE-SU) or recall based metric because we did not set a limit for either the number of chunks or the total length of the text returned for each task.

[20] We used `/resources/stopwords-rouge-default.txt` from ROUGE 2.0 tool. `http://kavita-ganesan.com/content/rouge-2.0`.

[21] `http://nlp.stanford.edu/software/corenlp.shtml`, Ver. 3.5.2

[22] `http://mallet.cs.umass.edu/`, Ver. 2.0.7

[23] `http://www.csie.ntu.edu.tw/~cjlin/liblinear/`, Ver. 1.8

Table 2: Evaluation results for search task suggestion (STS) and automatic procedural knowledge base construction (APKBC). A dagger (†) and one to three stars (⋆) represent significance levels of $p \leq 0.1$, 0.05, 0.01, and 0.001 respectively, and ns represents not significant.

	Classifiers					W/ subsets of features (+context)				W/O subsets of features (+context)				Local	Context
	CRF	HMM	SVM	LR	TFIDF	POS	PAR	LOC	WORD	POS	PAR	LOC	WORD		
STS															
Macro P	.8691	.5281***	.7589***	.8461^{ns}	.0087***	.8346†	.6846***	.2476***	.8164**	**.8811**†	.8689^{ns}	.8431**	.8699^{ns}	.7808**	.8155**
Macro R	**.6563**	.1984***	.6428†	.5728***	.0072***	.4720***	.3245***	.0900***	.5849***	.6493^{ns}	.6519^{ns}	.6470*	.5133***	.5935**	.6248**
Macro F1	**.7471**	.2870***	.6955**	.6826***	.0079***	.6015***	.4390***	.1241***	.6803***	.7466^{ns}	.7442^{ns}	.7311**	.6444***	.6737**	.7068**
Micro P	.9359	.9160†	.8641***	.9142*	.4431***	.9248^{ns}	.8811**	.5980*	.9105**	.9400^{ns}	.9357^{ns}	.9153**	**.9411**^{ns}	.8935**	.8983**
Micro R	**.6934**	.3113***	.6674*	.6078***	.0262***	.5487***	.3973***	.1671***	.6185***	.6856†	.6864^{ns}	.6800*	.5867***	.6449**	.6538**
Micro F1	**.6930**	.3161***	.6612**	.6144***	.0244***	.5541***	.4018***	.1678***	.6175***	.6870†	.6861†	.6785*	.5911***	.6447**	.6535**
ROUGE-2	**.8112**	.5200***	.7922*	.7476**	.2279***	.7108***	.6020***	.3488***	.7713**	.8113^{ns}	.8069†	.8104^{ns}	.7322**	.7807**	.7907*
ROUGE-S4	**.8087**	.5147***	.7892*	.7446**	.2041***	.7079***	.5954***	.3420***	.7657**	.8082^{ns}	.8044†	.8065^{ns}	.7299**	.7746**	.7857**
APKBC: summary															
Macro P	.7576	.0042***	.1464***	.1131***	.0160***	.5325*	.3175**	-	**.7738**^{ns}	.7489^{ns}	.7723^{ns}	-	.5359*	.7590^{ns}	.7736†
Macro R	**.3260**	.0008**	.1116**	.0489**	.1254*	.2881^{ns}	.0641**	-	.2464**	.2651**	.3167^{ns}	-	.2945^{ns}	.2757*	.3113†
Macro F1	**.4207**	.0014**	.1175***	.0668**	.0271**	.3556^{ns}	.0988**	-	.3279**	.3460**	.4129^{ns}	-	.3632^{ns}	.3755*	.4081^{ns}
Micro P	.8524	.1213***	.4585***	.4210**	.0556***	.7093**	.7893†	-	**.9061***	.8813^{ns}	.8749†	-	.7084**	.8439^{ns}	.8604^{ns}
Micro R	**.3344**	.0010**	.1041**	.0476**	.1347*	.2923^{ns}	.0838**	-	.2433**	.2576**	.3088†	-	.3185^{ns}	.2734*	.3159^{ns}
Micro F1	**.3455**	.0015**	.1119***	.0511**	.0683**	.3153^{ns}	.1996**	-	.2526**	.2695**	.3198†	-	.3408^{ns}	.2902*	.3260^{ns}
ROUGE-2	**.4463**	.1017**	.2425**	.1734**	.1583**	.3822^{ns}	.1996*	-	.3435**	.3678**	.4170*	-	.3787^{ns}	.4053*	.4069*
ROUGE-S4	**.4392**	.0660**	.2301**	.1692**	.1026**	.3788^{ns}	.1968*	-	.3412**	.3645**	.4118*	-	.3704^{ns}	.4006*	.4022*
APKBC: explanation															
Macro P	.4984	.0000***	.0053***	.0000***	.0002***	.1904**	**.7000**^{ns}	-	.4901^{ns}	.4722†	.4837^{ns}	-	.2546**	.1337***	.4534^{ns}
Macro R	**.3310**	.0000**	.0026**	.0000**	.0015**	.0463**	.0000**	-	.3056†	.3198^{ns}	.3169†	-	.0247**	.0355**	.2879*
Macro F1	**.3853**	.0000***	.0034***	.0000***	.0003***	.0687**	.0000***	-	.3639†	.3695^{ns}	.3718^{ns}	-	.0410**	.0521***	.3359*
Micro P	.7180	.4724*	.4911**	.6978^{ns}	.0062***	.7588^{ns}	**.9909*****	-	.7292^{ns}	.7206^{ns}	.7274^{ns}	-	.7575^{ns}	.6105^{ns}	.6823^{ns}
Micro R	.3342	.0050**	.0012**	.0000**	.0363**	.0704**	.0000**	-	.2961^{ns}	**.3411**^{ns}	.3208^{ns}	-	.0194**	.0370**	.2723^{ns}
Micro F1	**.3577**	.0050**	.0018**	.0000**	.0103**	.0889**	.0000**	-	.3176^{ns}	.3554^{ns}	.3468^{ns}	-	.0292**	.0474**	.2922^{ns}
ROUGE-2	.3698	.2450*	.1737**	.1470**	.2657†	.1647**	.1325**	-	.3489^{ns}	.3580^{ns}	**.3804**^{ns}	-	.1614**	.1858**	.3532^{ns}
ROUGE-S4	.3686	.2324*	.1659**	.1408**	.2498*	.1637**	.1325**	-	.3472^{ns}	.3554^{ns}	**.3793**^{ns}	-	.1614*	.1828**	.3500^{ns}

ducted a t-test to calculate the significance level of each baseline method compared against the proposed approach (CRF). We can see the proposed CRF-based sequence labeling approach can significantly outperform other baseline classifiers (at a significance level of ≤ 0.05 in terms of F-1 and ≤ 0.1 in terms of ROUGE).

The performance gap between the sequence labeling approaches and the independent classification-based approaches such as LR and SVM suggests that the query construction problem is similar to other sequence labeling problems such as named entity detection or supervised key word extraction. With only surface form features, HMM performs worse than CRF. All the supervised approaches, which take advantage of the constructed parallel corpus, can outperform the unsupervised approach (TF-IDF).

When we test each feature category (POS, Parsing, Location, Word, Local, and Context) independently, we can see none of them can beat the proposed approach with a confidence level of ≥ 0.99. We also see that word features contribute the most to performance, which can achieve around 90% of the performance in terms of F-1 and 95% in terms of ROUGE when all the features are used. Both F-1 and ROUGE scores drop when we remove any feature category from the feature list, and they drop the most when Word features are removed, which implies that all the features have positively contributed to the query construction task.

To better understand what non-Word features most likely promote a word to become a $\texttt{B}^{\texttt{Q}}$ or $\texttt{I}^{\texttt{Q}}$, we shows the top-5 features for $\texttt{O} \rightarrow \texttt{I}^{\texttt{Q}}$, $\texttt{B}^{\texttt{Q}} \rightarrow \texttt{I}^{\texttt{Q}}$, $\texttt{I}^{\texttt{Q}} \rightarrow \texttt{I}^{\texttt{Q}}$ transitions in Table 3. We see that the whole query phrase is very likely extracted from the summary part of a description (sum) due to its clarity and conciseness. We also see that both singular nouns (NN or NNP) and verbs in either VB or VBP forms are selected as indicators for beginning a query, and verb phase (VP) is a useful feature when deciding whether to continue a query.

Table 4: End-to-end evaluation results for search task suggestion (STS) and title (TI) and explanation (EX) generation for automatic procedural knowledge base construction (APKBC). PROP/P. corresponds to the proposed approach and WH corresponds to the wikiHow knowledge.

	STS				APKBC		
	PROP	GOOG	BING	LOG	P. TI	P. EX	WH
Macro P	**.4457**	.0972	.0333	.0676	**.0997**	.2046	**.9677**
Micro P	**.4457**	.0973	.0313	.0612	**.0995**	.2041	**.9515**
MAP	**.3361**	.0553	.0120	.0549	**.0527**	.1331	**.9404**

We then evaluated the end-to-end performance of the proposed STS solution using the query construction model. We present the results in Table 4 and show examples in Table 5. We see the proposed solution can outperform the commercial search engines by +0.35 to +0.41 in terms of Precision@8 and +0.28 to +0.32 in terms of MAP. We see that our proposed method, which is tailored for task-oriented search, can provide unique insights when suggesting tasks, compared to current general-purpose commercial search engines, which have been designed for entity-centric search and tend to suggest queries by appending keywords such as "product", "image", "logo", "online", "free", etc. We discovered three types of errors from our system: (1) Some suggested queries are ambiguous when presented by themselves (e.g. "fix it", "install"). (2) duplicated queries are suggested from different subtasks. (3) the manually created knowledge sometimes still contains a few non-instructional "steps". We could further improve the proposed approach by conducting a co-reference analysis or other approaches to incorporate original queries and contexts in the suggested queries. We could also collectively rank the candidates to avoid suggesting duplicated tasks.

Table 3: The non-Word features that contribute the most to each label in both search task suggestion (STS) and automatic procedural knowledge base construction (APKBC) problems. "+1"/"-1" presents the feature of the next/previous word, "sum" refers to summary, "VP"/"NP" represents if the word is inside a verb/noun phrase, "Begin" refers to the beginning of the text.

	STS ($M^{\mathtt{Q}}$)			APKBC: summary ($M^{\mathtt{TS}}$)			APKBC: explanation ($M^{\mathtt{TE}}$)		
	$\mathtt{O} \rightarrow \mathtt{B}^{\mathtt{Q}}$	$\mathtt{B}^{\mathtt{Q}} \rightarrow \mathtt{I}^{\mathtt{Q}}$	$\mathtt{I}^{\mathtt{Q}} \rightarrow \mathtt{I}^{\mathtt{Q}}$	$\mathtt{O} \rightarrow \mathtt{B}^{\mathtt{TS}}$	$\mathtt{B}^{\mathtt{TS}} \rightarrow \mathtt{I}^{\mathtt{TS}}$	$\mathtt{I}^{\mathtt{TS}} \rightarrow \mathtt{I}^{\mathtt{TS}}$	$\mathtt{O} \rightarrow \mathtt{B}^{\mathtt{TE}}$	$\mathtt{B}^{\mathtt{TE}} \rightarrow \mathtt{I}^{\mathtt{TE}}$	$\mathtt{I}^{\mathtt{TE}} \rightarrow \mathtt{I}^{\mathtt{TE}}$
1	*POS*: NNP	*POS*$^{-1}$: VB	*LOC*: sum	*POS*: VB	*POS*$^{-1}$: VB	*POS*$^{-1}$: VBP	Begin	VP	*POS*$^{-1}$: NN
2	*LOC*: sum	*LOC*: sum	*POS*$^{-1}$: IN	*POS*: VBP	*POS*$^{-1}$: VBP	*POS*: NNP	*POS*: VBG	*POS*$^{-1}$: NN	VP
3	*DEP*: ccomp	*POS*$^{-1}$: VBP	VP	*POS*: NN	*POS*$^{-1}$: NNP	*POS*$^{-1}$: IN	*POS*: NN	*POS*$^{-1}$: DT	*POS*$^{-1}$: NNS
4	*POS*: VB	*POS*$^{-1}$: NNP	*DEP*: dobj	*DEP*: appos	*POS*$^{-1}$: NN	*DEP*: xcomp	*DEP*: compound	NP	*POS*$^{-1}$: ,
5	*DEP*: nsubjpass	*POS*$^{-1}$: NN	*POS*$^{+1}$: JJ	*POS*: NNP	*DEP*: case	*POS*: JJR	*DEP*: nsubj	*POS*$^{-1}$: VB	*POS*$^{-1}$: NNP

Table 5: Comparison of top 4 suggested queries produced by the proposed STS system and commercial search engines for example tasks.

PROPOSED	GOOGLE	BING
Task: make blueberry banana bread		
blend the mixture	vegan blueberry banana bread	best blueberry banana bread
mix flour	buttermilk blueberry banana bread	bisquick banana blueberry bread
add 1 egg	great blueberry banana bread recipe	banana blueberry cake
vanilla extract	blueberry banana bread with yogurt	healthy banana blueberry bread
Task: slim down		
weight loss	slim down diet	the slim down club
heavy food	7 day slim down	how to slim down fast
junk food	weight loss	slim down challenge
keep up the mood	slim down thighs	how to slim down legs
Task: play red alert2		
build a barracks	red alert 2 complete (iso) original 2 disc	play red alert 2 game
build a war factory	play red alert 2 free	play ra2 online
radar chould	play red alert 2 online free	red alert 2 download
build a power plant/tesla reactor	play red alert 3	free red alert 3

Table 6: Partial suggested tasks from the top 8 outputs produced by the proposed APKBC system for example tasks.

Task: sign up for airbnb
Airbnb is no longer running the $50 OFF $200 promo but you can still save $25 OFF Your First Airbnb Stay of $75 or more by copying and pasting this link into your browser: ...
Task: make blueberry banana bread
Please don't use regular whole wheat in this recipe – the loaf will turn out very dense
Add the wet ingredients - the egg mixture to the flour mixture and stir with a rubber spatula until just combined
If you're in need of a quick, easy and delicious way to use up the ripe bananas in your house... definitely
Task: become a cell phone dealer
However, the cell phone provider may place restrictions on the manner in which you can use its company name, phone brands and images
Visit the state's business licensing agency's website and your city's occupational/business licensing department's website to determine if you need a license for your prepaid cell phone business

5.5 Automatic procedural knowledge base construction result

We compare the results of our proposed approach with the baseline methods for task description (title and explanation) construction in Table 2. We first see that all the methods produce much lower F-1 and ROUGE scores for APKBC than those for STS, which suggests that task description generation is a difficult task, esp. explanation construction. Nevertheless, we can see that the proposed approach significantly outperforms all other classifiers in terms of F-1 and ROUGE metrics (at a significance level of ≤ 0.1).

When comparing with the results from using each feature category, we see that not a single category can reach the same performance level as the proposed approach, which again implies that all the feature categories have contributed to the overall performance in task description construction problem. Comparing with using all but one feature categories, we can find that Word features are crucial for summary generation but not explanation generation, whereas POS and Parsing features are crucial for explanation generation but not summary generation. We also observe the non-Word features that contribute the most in Table 3. Although the performance does not match that of the query construction approach, we can still find interesting patterns in the results. First, we can see that nouns and verbs are still crucial in constructing task descriptions, and verbs (VB and VBP) rather than nouns are more likely selected as the beginning of a summary. To begin the task explanation, the word is expected to have a "Begin" indicator and/or a dependency type of nsubj, meaning the nominal subject of a sentence. Verb phrases are also important in identifying a task description.

Finally, we evaluated our proposed end-to-end APKBC solution, and compared it with wikiHow articles in Table 4. Due to the difficulty of the task description construction problem and lack of high-quality task-oriented search query candidates, we find that the automatic approach performs much worse than manual curation in building a brand new procedural knowledge base from scratch; nevertheless, we still find that the proposed approach is able to discover relevant subtasks that are not covered in the current wikiHow article, i.e. it can be used to improve the coverage and quality of existing procedural knowledge bases. Some examples are shown in Table 6. For example, for the task "sign up for airbnb", one of the suggested queries "sign up for airbnb coupon" implies a coupon may be an important subsidiary resource of a task that the current wikiHow article does not know. For this particular task, a statement with the detailed coupon info is able to be extracted to describe the concrete task related to the resource. Even though this result may be from an advertisement Web page, it still delivers the freshest information that can hardly be added and updated instantly in manually created procedural knowledge bases.

We also summarize the errors into three major categories: (1) ambiguous task representations. For example, the "find a match" article on wikiHow[24] describes how to "look for a partner", whereas search engines try to disambiguate the search intent by also suggesting queries in the context of data analysis such as "find a match in excel", "find a match spreadsheet", etc., (2) duplicated descriptions, and (3) quality of the text extracted from noisy Web documents. Moreover, as we note that not all wikiHow articles exactly follow the writing guide, the proposed approach can be used estimate the quality of each existing task description, similar to Eq. 3, and report potentially suspicious low-quality articles or spurious steps such as "You're done.", which violate the writing guidelines.

[24] http://www.wikihow.com/Find-a-Match

6. CONCLUSION

This paper provides a first attempt to bridge the gap between the two evolving research areas: construction of procedural knowledge bases and task-oriented search. We investigated two problems: *search task suggestion using procedural knowledge* and *automatic procedural knowledge base construction from search activities*. We proposed the creation of a three-way parallel corpus of queries, query contexts, and task descriptions, and reduced both problems to sequence labeling tasks. We proposed a supervised framework to identify the queryable phrases from task descriptions and extract procedural knowledge from search queries and relevant text snippets. We compared our proposed solution with baseline algorithms, and used commercial search engines and manually-curated procedural knowledge for both problems; experimental results show an improvement of +0.28 to +0.41 in terms of Precision@8 and MAP.

As a next step, we intend to conduct a user study to estimate to what extent the users can find good tips for real-world information needs. We also plan to further study how to properly rank the candidate queries and task descriptions, and also study the APKBC problem by leveraging a supervised natural language generation framework in addition to the proposed sequence labeling approach.

7. REFERENCES

[1] A. Addis and D. Borrajo. From unstructured web knowledge to plan descriptions. In *DART'2009*, volume 324, pages 41–59, 2011.

[2] J. R. Anderson. *Cognitive psychology and its implications* . WH Freeman/Times Books/Henry Holt & Co, 1990.

[3] R. Baeza-Yates, C. Hurtado, and M. Mendoza. Query recommendation using query logs in search engines. In *Proceedings of EDBT'2004*, pages 588–596, 2005.

[4] S. Banerjee and S. Bandyopadhyay. Question classification and answering from procedural text in english. In *Proceedings of QACD'2012*, pages 11–26, 2012.

[5] I. Bordino, G. De Francisci Morales, I. Weber, and F. Bonchi. From machu_picchu to rafting the urubamba river: anticipating information needs via the entity-query graph. In *Proceedings of WSDM'2013*, pages 275–284, 2013.

[6] I. Bordino, Y. Mejova, and M. Lalmas. Penguins in sweaters, or serendipitous entity search on user-generated content. In *Proceedings of CIKM'2013*, pages 109–118, 2013.

[7] C. Brewster, K. O'Hara, S. Fuller, Y. Wilks, E. Franconi, M. A. Musen, J. Ellman, and S. Buckingham Shum. Knowledge representation with ontologies: the present and future. *IEEE Intell. Syst.*, pages 72–81, 2004.

[8] A. Broder. A taxonomy of web search. In *ACM SIGIR forum*, volume 36, pages 3–10, 2002.

[9] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li. Context-aware query suggestion by mining click-through and session data. In *Proceedings of SIGKDD'2008*, pages 875–883, 2008.

[10] N. Chambers and D. Jurafsky. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08: HLT*, pages 789–797, 2008.

[11] Z. Cheng, B. Gao, and T.-Y. Liu. Actively predicting diverse search intent from user browsing behaviors. In *Proceedings of WWW'2010*, pages 221–230, 2010.

[12] N. Craswell and M. Szummer. Random walks on the click graph. In *Proceedings of SIGIR'2007*, pages 239–246, 2007.

[13] H. K. Dai, L. Zhao, Z. Nie, J.-R. Wen, L. Wang, and Y. Li. Detecting online commercial intention (oci). In *Proceedings of WWW'2006*, pages 829–837, 2006.

[14] J. Dalton, L. Dietz, and J. Allan. Entity query feature expansion using knowledge base links. In *Proceedings of SIGIR'2014*, pages 365–374, 2014.

[15] N. Dalvi, R. Kumar, B. Pang, R. Ramakrishnan, A. Tomkins, P. Bohannon, S. Keerthi, and S. Merugu. A web of concepts. In *Proceedings of PODS'2009*, pages 1–12, 2009.

[16] E. Delpech and P. Saint-Dizier. Investigating the structure of procedural texts for answering how-to questions. In *Proceedings of LREC'2008*, pages 46–51, 2008.

[17] Y. Fukazawa and J. Ota. Automatic modeling of user's real world activities from the web for semantic ir. In *Proceedings of SemSearch'2011*, pages 5:1–5:9, 2010.

[18] M. P. Georgeff and A. L. Lansky. Procedural knowledge. *Proceedings of the IEEE*, 74(10):1383–1398, 1986.

[19] Q. Guo and E. Agichtein. Ready to buy or just browsing?: detecting web searcher goals from interaction data. In *Proceedings of SIGIR'2010*, pages 130–137, 2010.

[20] A. Hassan and R. W. White. Task tours: helping users tackle complex search tasks. In *Proceedings of CIKM'2012*, pages 1885–1889, 2012.

[21] A. Hassan Awadallah, R. W. White, P. Pantel, S. T. Dumais, and Y.-M. Wang. Supporting complex search tasks. In *Proceedings of CIKM'2014*, pages 829–838, 2014.

[22] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *Proceedings of WWW'2006*, pages 387–396, 2006.

[23] Y. Jung, J. Ryu, K.-m. Kim, and S.-H. Myaeng. Automatic construction of a large-scale situation ontology by mining how-to instructions from the web. *Web Semant.*, 8(2–3):110–124, 2010.

[24] M. P. Kato, T. Yamamoto, H. Ohshima, and K. Tanaka. Investigating users' query formulations for cognitive search intents. In *Proceedings of SIGIR'2014*, pages 577–586, 2014.

[25] U. Lee, Z. Liu, and J. Cho. Automatic identification of user goals in web search. In *Proceedings of WWW'2005*, pages 391–400, 2005.

[26] Y. Li, B.-J. P. Hsu, and C. Zhai. Unsupervised identification of synonymous query intent templates for attribute intents. In *Proceedings of CIKM'2013*, pages 2029–2038, 2013.

[27] T. Lin, P. Pantel, M. Gamon, A. Kannan, and A. Fuxman. Active objects: Actions for entity-centric search. In *Proceedings of WWW'2012*, pages 589–598, 2012.

[28] J. Liu, M. J. Cole, C. Liu, R. Bierig, J. Gwizdka, N. J. Belkin, J. Zhang, and X. Zhang. Search behaviors in different task types. In *Proceedings of JCDL'2010*, pages 69–78, 2010.

[29] Y. Liu, R. Song, M. Zhang, Z. Dou, T. Yamamoto, M. Kato, H. Ohshima, and K. Zhou. Overview of the ntcir-11 imine task. In *Proceedings of NTCIR-11*, pages 8–23, 2014.

[30] G. Marchionini. Exploratory search: from finding to understanding. *Commun. ACM*, 49(4):41–46, 2006.

[31] A. K. McCallum. Mallet: A machine learning for language toolkit. 2002.

[32] Q. Mei, D. Zhou, and K. Church. Query suggestion using hitting time. In *Proceedings of CIKM'2008*, pages 469–478, 2008.

[33] K. L. Myers. A procedural knowledge approach to task-level control. In *AIPS'1996*, pages 158–165, 1996.

[34] P. Pareti, E. Klein, and A. Barker. A semantic web of know-how: Linked data for community-centric tasks. In *Proceedings of WIC'2014*, pages 1011–1016, 2014.

[35] G. Pass, A. Chowdhury, and C. Torgeson. A picture of search. In *InfoScale*, volume 152, page 1, 2006.

[36] M. Perkowitz, M. Philipose, K. Fishkin, and D. J. Patterson. Mining models of human activities from the web. In *Proceedings of WWW'2004*, pages 573–582, 2004.

[37] F. Radlinski, M. Szummer, and N. Craswell. Inferring query intent from reformulations and clicks. In *Proceedings of WWW'2010*, pages 1171–1172, 2010.

[38] D. E. Rose and D. Levinson. Understanding user goals in web search. In *Proceedings of WWW'2004*, pages 13–19, 2004.

[39] R. C. Schank and R. P. Abelson. *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*. Psychology Press, 2013.

[40] I. Weber, A. Ukkonen, and A. Gionis. Answers, not links: Extracting tips from yahoo! answers to address how-to web queries. In *Proceedings of WSDM'2012*, pages 613–622, 2012.

[41] R. W. White and J. Huang. Assessing the scenic route: measuring the value of search trails in web logs. In *Proceedings of SIGIR'2010*, pages 587–594, 2010.

[42] Z. Yang, Y. Li, J. Cai, and E. Nyberg. Quads: Question answering for decision support. In *Proceedings of SIGIR'2014*, pages 375–384, 2014.

[43] X. Yu, H. Ma, B.-J. P. Hsu, and J. Han. On building entity recommender systems using user click log and freebase knowledge. In *Proceedings of WSDM'2014*, pages 263–272, 2014.