An Empirical Study on Categorizing User Input Parameters for User Inputs Reuse

Shaohua Wang[1], Ying Zou[2], Bipin Upadhyaya[2], Iman Keivanloo[2], Joanna Ng[3]

[1] School of Computing, Queen's University, Kingston, Ontario, Canada
shaohua@cs.queensu.ca
[2] Electrical and Computer Engineering, Queen's University, Kingston, Canada
{ying.zou, bipin.upadhyaya, iman.keivanloo}@queensu.ca
[3] CAS Research, IBM Canada Software Laboratory, Markham, Ontario, Canada
{jwng}@ca.ibm.com

Abstract. End-users often have to enter the same information to various services (*e.g.*, websites and mobile applications) repetitively. To save end-users from typing redundant information, it becomes more convenient for an end-user if the previous inputs of the end-user can be pre-filled to applications based on end-user's contexts. The existing pre-filling approaches have poor accuracy of pre-filling information, and only provide limited support of reusing user inputs within one application and propagating the inputs across different applications. The existing approaches do not distinguish parameters, however different user input parameters can have very varied natures. Some parameters should be pre-filled and some should not. In this paper, we propose an ontology model to express the common parameters and the relations among them and an approach using the ontology model to address the shortcomings of the existing pre-filling techniques. The basis of our approach is to categorize the input parameters based on their characteristics. We propose categories for user inputs parameters to explore the types of parameters suitable for pre-filling. Our empirical study shows that the proposed categories successfully cover all the parameters in a representative corpus. The proposed approach achieves an average precision of 75% and an average recall of 45% on the category identification for parameters. Compared with a baseline approach, our approach can improve the existing pre-filling approach, *i.e.*, 19% improvement on precision on average.

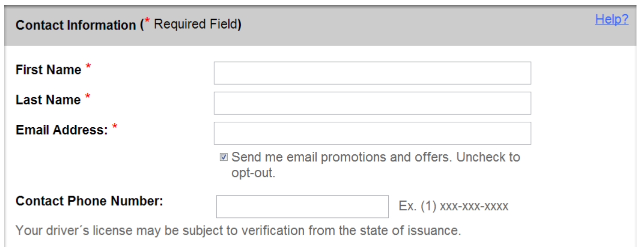
Keywords: User Input Parameters Categories; User Inputs Reuse; Auto-filling; Ontology; Web Forms;

1 Introduction

Web is becoming an essential part of our daily life. Various on-line services (*e.g.*, web services and mobile applications) allow end-users to conduct different web tasks, such as on-line shopping and holiday trip planning. These services require end-users to provide data to interact with them and some of the data provided by end-users are usually repetitive. For example the AVIS[4], a car rental

[4] http://www.avis.ca/car-rental/avisHome/home.ac

website, and the Homes [5], a real estate website, require end-users to enter their personal information such as *First Name* and *Last Name* illustrated in Figure 1 and Figure 2. It could be a cumbersome and annoying process for an end-user, especially a smartphone end-user, to fill the same information into web forms with multiple input fields repeatedly. Therefore information pre-filling becomes critical to save end-users from this cumbersome process. Rukzio et al. [1] found that end-users are four times faster on smartphones when they just have to correct pre-filled form entries compared to entering the information from scratch. A web form within web or mobile applications usually consists of a set of input fields assigned with a label, for example the input field *Contact Phone Number* in Figure 1. The label of this input field is "Contact Phone Number", the type of this input field is *text field*. Throughout this paper, we consider an input field as a user input parameter and an end-user input (*i.e.*, a piece of information) as the value which can be filled into an input field.


Contact Information (* Required Field) Help?

First Name *

Last Name *

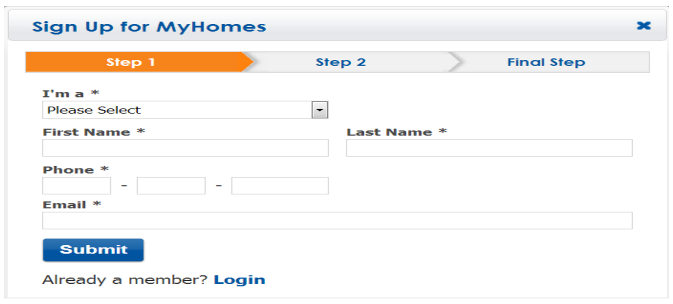
Email Address: *

☑ Send me email promotions and offers. Uncheck to opt-out.

Contact Phone Number: Ex. (1) xxx-xxx-xxxx

Your driver's license may be subject to verification from the state of issuance.

Fig. 1: A sample screen shot of a web form requiring user's personal information to reserve a car.


Sign Up for MyHomes

Step 1 Step 2 Final Step

I'm a *

Please Select

First Name * Last Name *

Phone *

Email *

Submit

Already a member? Login

Fig. 2: A sample screen shot of a web form requiring user's personal information to sign up the website.

Recently, several industrial tools and academic approaches have been developed to pre-fill user inputs into web forms automatically. Web browsers usually provide web form Auto-filling tools, such as Firefox form auto-filling Addon [2] and Chrome form auto-filling tool [3] to help users fill in forms. In general, the auto-filling tools record the values entered by a user for specific input fields pre-configured by the tools in a given form. Then, they identify the input fields

[5] http://www.homes.com/

which are identical or similar to the pre-configured input fields by string matching, and fill the entered values into the identified inputs fields when the users visit websites. Since such approaches are error-prone, they allow end-users to modify the values. Recently, a few academic studies such as [4][5][6][7] proposed several approaches to help end-users. Hartman et al. [4] propose a context-aware approach using user's contextual information to fill in web forms. Toda et al. [5] propose a probabilistic approach using the information extracted from data-rich text as the input source to fill values to web forms. Wang et al. [6] propose an approach using the relations between similar input parameters to fill values to web forms. Araujo et al. [7] extract the semantic of concepts behind the web form fields and use the relations of concepts to fill in web forms. However, all these tools and approaches suffer from two main drawbacks:

– **Poor accuracy of pre-filling values to input parameters of web forms or applications**. The pre-filled information can be out of date and out of context. The user often has to modify the pre-filled values passively. With the rapid growth of the population of mobile application users, it is even more frustrating for mobile application users to modify the pre-filled values due to the restrictions in screen size and typing. Accurate pre-filling values becomes a critical step to enhance the user experience.
– **Limited support of reusing user inputs within an application and propagating them across different applications.** The existing methods can only reuse a few types of user inputs within an application or across different applications. For example an end-user is planning a holiday trip from Toronto to Miami, he or she could conduct two tasks of searching for cheap flight tickets and cheap transportation from airport to hotel. The end-user needs browse different websites to find the most appropriate solution for him or her. During the process of conducting these two tasks, the similar or identical input parameters from different websites or within a website, such as departure and return cities, should be linked together, and reuse user inputs among them. In this scenario, for example, Chrome form auto-filling tool [3] cannot help the end-users, since it can only reuse basic personal information such as credit card information and addresses.

The existing tools and approaches treat all the user input parameters equally, and the matching mechanism of existing approaches is only based on the string matching or semantic similarity calculation of field names. If a match is identified, the existing approaches pre-fill a value to an input parameter. However different user input parameters can have very varied natures. For example, the coupon number is only valid for a single purchase, therefore may not be suitable for pre-filling. The input fields for departure and destination cities from a flight booking website should not be pre-filled with previous user inputs without knowing user's context, the end-user's name can be pre-filled, since the end-user's name does not change in most cases.

It is crucial to improve the accuracy of automatic value pre-filling by understanding the characteristics of different user input parameters. In this paper, we propose an ontology model for expressing common parameters and approach based on the proposed ontology model to automatically identify a category for an input parameter. We conducted two empirical studies. The first empirical study was served as an initial empirical investigation to explore the characteristics of user input parameters. The first empirical study was conducted on 30 popular websites and 30 Android mobile applications from Google Play Android Market [6]. Based on the results of our first study, we identify and propose four categories for input parameters from web and mobile applications. The proposed categories offer a new view for understanding input parameters and provide tool developers guidelines to identify pre-fillable input parameters and the necessary conditions for pre-filling. To the best of our knowledge, we are the first to propose categories for input parameters to explore the characteristics of user input parameters. We conduct the second empirical study to verify the effectiveness of proposed categories and approach for category identification. The second study was conducted on 50 websites from three domains and 100 mobile applications from five different categories in Google Play Android Market.

The major contributions of our paper are listed as follows:

- We propose four categories of user input parameters to capture the nature of different user input parameters through an empirical investigation. For pre-filling, each category of parameters should be collected, analyzed and reused differently. The results of our empirical study show that our categories are effective to cover all the input parameters in a representative corpus.
- We propose an ontology model to express the common parameters and the relations among them. Moreover, we use the proposed ontology model to carry the category information for input parameters. We propose a WordNet-based approach that automatically updates the core ontology for unseen parameters from new applications. The results of our empirical study show that our approach obtains a precision of 88% and a recall of 64% on updating the ontology to include the unseen parameters on average.
- We propose an ontology-based approach to identify a category for input parameters automatically. On average, our approach for category identification can achieve a precision of 90.5% and a recall of 72.5% on the identification of a category for parameters on average.
- We test the effectiveness of our proposed categories on improving the existing approaches. We build a baseline approach which does not distinguish the different characteristics of input parameters, and incorporate our proposed categories with the baseline approach to form a category-enabled approach. We compare two approaches through an empirical experiment. The results show that our approach can improve the baseline approach significantly, *i.e.*, on average 19% in terms of precision.

The rest of the paper is organized as follows. Section2 describes the user interfaces of web and mobile applications. Section 3 presents our proposed approach for categorizing input parameters. Section 4 introduces the empirical studies. Section 5 discusses the threats to validity. Section 6 summarizes the related literature. Finally, section 7 concludes the paper and outlines some avenues for future work.

[6] https://play.google.com/store?hl=en

2 Web and Mobile Application User Interface

In this paper, we study the user input parameters from the user interfaces of web applications and mobile applications.

Web pages are mainly built with HTML and Cascading Style Sheets (CSS), and can be converted into an HTML DOM tree [7]. A Web form is defined by an HTML FORM tag <form> and the closing tag </*form*>. A web form usually consists of a set of input elements, such as the text fields in Figure 2, to capture user information. An input element is defined by an HTML INPUT tag <input> specifying an input field where the user can enter data. The input element can contain several attributes, such as *name* specifying the name of an <input> element, *type* defining the type <input> to display (*e.g.*, displayed as a button or checkbox) and *value* stating the value of an <input> element. An <input> element can be associated with a human-readable label, such as First Name. This information can be accessible by parsing HTML source code.

There are three types of mobile applications:

- *Native Apps:* The native apps are developed specifically for one platform such as Android[8], and can access all the device features such as camera.
- *Web Apps:* The web apps are developed using standard Web technologies such as Javascript. They are really websites having *look and feel* like native apps. They are accessed through a web browser on mobile devices.
- *Hybrid Apps:* The hybrid apps are developed by embedding HTML5 apps inside a thin native container.

An Android application is encapsulated as an Android application package file (APK) [9]. An APK file can be decoded into a nearly original form which contains resources such as source code (*e.g.*, Java) and user interface layout templates in XML files that define a user interface page or a user interface component if it is not a HTML5 application. In this study, we only study the native mobile applications because the user interface templates in XML files can be obtained by decoding APK files.

3 Our Proposed Approach for Categorizing User Input Parameters

In this section, we first present an ontology model for expressing common user input parameters and their relations. Second, we propose an automatic approach for updating the ontology. Third, we propose an ontology-based approach for categorizing input parameters.

3.1 An Ontology Definition Model

In this study, we build an ontology to capture the common user input parameters and the five relations among parameters. Figure 3 illustrates the main components of ontology definition model and their relations.

[7] http://www.w3schools.com/htmldom/dom_nodes.asp
[8] http://www.android.com/
[9] http://en.wikipedia.org/wiki/APK_(file_format)

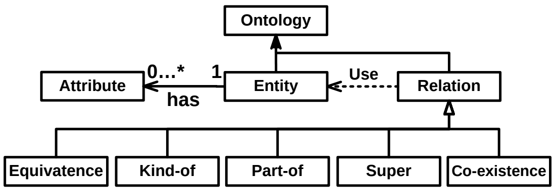


Fig. 3: Components of ontology definition model

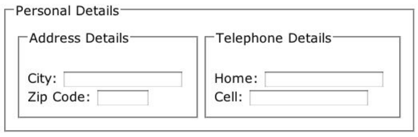


Fig. 4: An example of 4 user inputs parameters: City, Zip Code, Home and Cell

The components are listed as follows:

- *Entity:* is an input parameter from a website or mobile application, or a concept description of a group of resources with similar characteristics.
- *Attribute:* is a property that a parameter can have. There are four attributes:
  - *Category.* The category defines the category information of a parameter.
  - *Label.* The category stores the *Label* (*i.e.*, a human-readable description) of an input field from websites or mobile applications.
  - *Coding Information.* The category stores the HTML coding information (*i.e.*, websites) or XML templates (*i.e.*, mobile applications).
  - *Concepts.* The category stores the concepts related to the parameter.
- *Relation:* defines various ways that entities can be related to one another. The five relations are listed as follows:
  - *Equivalence.* Two entities have the same concept such as Zip Code and Postal Code.
  - *Kind-of.* One entity is a kind-of another one. For example, Home (*i.e.*, home phone) is a kind of Telephone Details illustrated in Figure 4.
  - *Part-of.* One entity is a part-of another one, such as Zip Code and Address Details illustrated in Figure 4.
  - *Super.* One entity is a super of another. The super relation is the inverse of the Kind-of relation. For example, Telephone Details is a super of Home (*i.e.*, home phone) illustrated in Figure 4.
  - *Co-existence.* Two entities are both a part-of an entity and they are not in the relation of equivalence, such as City and Zip Code illustrated in Figure 4.

Usually the user input parameters are terminal concepts [8], such as *Phone Number* and *Price*, which have no sub-concepts. During the process of ontology creation, we use the UI structure and semantic meanings of the parameters to identify the relations. If two input elements have the same parent node in an HTML DOM tree, their relation is co-existence, and the relation between the two input parameters and their parent HTML DOM node is part-of. For example,

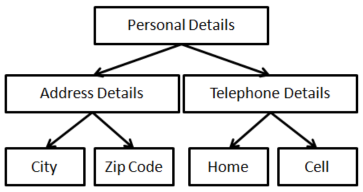


Fig. 5: An example ontology of personal details containing four user inputs parameters: City, Zip Code, Home and Cell

the user input parameters *City* and *Zip Code* co-exist and they have a part-of relation with Address Details in Figure 4. Figure 5 shows the visualization of the ontology that is built based on the example in Figure 4.

3.2 Our Approach of Updating Ontology

Once the initial ontology based on the proposed ontology model (Section 3.1) is established, an automatic approach for updating the ontology is required to add a new parameter into the ontology. We use WordNet [9], a large lexical database for English and containing semantic relations between words, to identify the relations between the new parameter and the existing ones in the ontology. The following relations of words defined in WordNet are used to identify our 5 relations:

1. If two words have the same synsets, they have a same semantic meaning, we convert it to Equivalence relation.
2. **Hypernym** shows a kind-of relation. For example, car is a hypernym of vehicle. We convert it to a kind-of relation.
3. **Meronym** represents a part-whole relation. We convert it to a part-of relation.
4. **Hyponym** defines that a word is a super name of another. We convert it to super relation. Hyponym is the inverse of hypernym meaning that a concept is a super name of another. For example, vehicle is a hyponym of car.
5. We use the part-of relation to identify the co-existence relation. If a word with another word both have a part-of relation with a same word, this word and the other word have co-existence relation.

3.3 Our Approach for Category Identification

It is important for a pre-filling approach to know the category of a user input parameter automatically. In this section, we introduce our ontology-based approach for identifying a category of an input parameter in details. Our approach uses the proposed ontology definition model proposed in Section 3.1. Our approach uses two strategies which are listed as follows:

– *Concept-based Strategy.* This strategy relies on an ontology of user input parameters to identify a category for a user input parameter automatically. If two input parameters have the same concepts, these two input parameters belong to the same category.

- *UI-based Strategy.* This strategy relies on the design of the UI layouts. We consider two user input parameters are the nearest neighbours to each other if their input fields are contained in the same parent UI component. Figure 4 shows an example of 4 user input parameters: *City, Zip Code, Home and Cell.* The *City and Zip Code* are the nearest neighbours to each other since they have the same parent UI node which has a label *Address Details.* We assume that if all the neighbours of a user input have been categorized and belong to the same category, there exists a high chance that they belong to the same category.

Given an ontology having a set of parameters, which is denoted as O = $< P_1^O, \ldots, P_n^O >$, where n is the number of parameters, and an input parameter P, our approach uses the following steps to identify a category for P:

- Step 1. We use WordNet synsets to retrieve synonyms of the concepts of the parameter P to expand the concept pool (*i.e.*, a bag of words) of P and the concept pool of each P_i^O, where $1 \leq i \leq n$.
- Step 2. We identify the P_j^O (where $1 \leq j \leq n$) whose concept pool has the concepts which are the synonyms of (or identical to) any concepts in the concept pool of P (*i.e.*, having the same semantic meaning).
- Step 3. If multiple parameters in the ontology are identified for P in Step 2, we choose the parameter sharing the most common concepts with the concept pool of P as the identical parameter to P. We assign the category of the chosen parameter to P
- Step 4. If no parameter is identified in Step 2, we apply the UI-strategy on the given ontology O and input parameter P. We identify the neighbors of P from its UI and repeat Step 1-3 to identify a category for every neighbor. If any neighbor of P cannot be categorized (*i.e.*, no parameters in the ontology have the same concepts as the neighbor does) or the neighbors of P have different categories, we cannot categorize P. If all the neighbors of P belong to a same category, we assign this category to input parameter P.

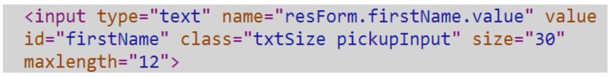
4 Empirical Study

In this study, we conduct two studies on different datasets. The first study is designed to study the different characteristics of parameters and propose categories for input parameters. The second empirical study is designed to evaluate the effectiveness of the proposed categories and the approach for categorizing input parameters.

4.1 First Empirical Study

We conduct an initial study to understand the nature of user input parameters and categorize the parameters. Understanding the different characteristics of parameters of different categories can help analyze, process and pre-fill the parameters differently to improve the accuracy of pre-filling. In this section, we introduce the study setup, the analysis method and the findings of our empirical investigation.

4.1.1 Data Collection and Processing. The study is conducted on 30 websites (*i.e.*, 15 shopping websites and 15 travel planning websites) and 30 mobile applications (*i.e.*, 15 shopping apps and 15 travel apps). We collect the user input parameters from web and mobile applications in different ways.



```
<input type="text" name="resForm.firstName.value" value
id="firstName" class="txtSize pickupInput" size="30"
maxlength="12">
```

Fig. 6: A sample screen shot of source code of an input field.

Collecting input parameters from websites: First, we manually visit the website to bypass the login step. Second, we use *Inspect Element*[10] of Google Chrome[11] to collect the following information of a user input parameter:

– **Label:** The value of the label (*i.e.*, the descriptive text shown to users).
– **Attributes:** The values of the attributes of the input element such as id and name. For example, Figure 6 shows the source code of the input field First Name in Figure 1.

Collecting input parameters from mobile applications: Instead of running each mobile application, we use Android-apktool [12], a tool for reverse engineering Android APK files, to decode resources in APKs to readable format. We build a tool to extract the information of a form from UI layout XML files if exist, then collect the information related to a user input parameter in the same way as we collect from websites.

Data cleaning: The extracted information need to be cleaned for further processing. For example, the extracted information for the input parameter *First Name* in Figure 1 is {First Name, resForm.firstName.value, FirstName, text} need to be cleaned. We remove the duplicated phrases and programming expressions. For the given example above, the output after cleaning is {First Name, res, Form, value, text}.

Data processing: We manually process the information of parameters as follows: First, we identify the concepts from the information of a user input parameter. A concept is a semantic notion or a keyword for describing a subject (*e.g.*, “taxi” and “city”). Second, the parameters having the same concepts or same semantical meanings are merged and considered as one parameter which we save all the unique concepts for. For example, two input parameters from two different flight searching websites, one with the label *Departure* and the other one with the label *Leave* should be merged and considered as one parameter having two concept words *Departure* and *Leave*.

[10] https://developers.google.com/chrome-developer-tools/docs/elements
[11] https://www.google.com/intl/en/chrome/browser/
[12] https://code.google.com/p/android-apktool/

4.1.2 Results of First Empirical Study. We collected 76 and 32 unique user input parameters from websites and mobile applications respectively. Due to the fact that the user input parameters are repeated among different applications, we observe that the number of unique parameters identified decreases with the increase of the number of applications studied. The 76 parameters extracted from the websites (*i.e.*, shopping and travel) contain all the 32 parameters from mobile applications from the same domains. This is due to the fact that mobile applications usually are the simplified versions of corresponding websites.

After studied the user input parameters, we found that input parameters can be categorized into four categories. The four categories are listed as follows:

- *Volatile Parameters:* This type of parameters have no change of state. They can be further categorized into two sub-categories:
  - One-Time Parameters: The values of this type of parameters are valid for one interaction, such as coupon number. This type of parameters should not be used for pre-filling at all.
  - Service-Specific Parameters: The values of this type of parameters can only be used for a specific service (*e.g.*, a website or a mobile application). For example, Member ID of the BESTBUY Reward Zone[13] in Canada can only be used for "Sign In" function or "Account Set Up" function of reward service. When end-users receive a membership card, they need enter the Member ID to set up an account in BESTBUY Reward Zone. This type of parameters can be pre-filled, however the parameters cannot be reused by different services.
- *Short-time Parameters:* The values of this type of parameters change with high frequency. For example during the course of conducting an on-line clothes shopping, an end-user may use the different colors as the search criteria to find the best suitable clothes from different services for the user, during a short period of time, the value of the color can be pre-filled. This type of parameters can be pre-filled and reused by different services, however it is extremely hard to pre-fill this type of parameters unless some conditions are met. For example, in the above example of searching clothes, the value of the color should be pre-filled only if the end-user has not switched to a new task.
- *Persistent Parameters:* The values of this type of parameters do not change over a long period of time. For example, the gender of the end-user and permanent home address. This type of parameters are suitable for pre-filling and being reused across different services.
- *Context-aware Parameters:* There exist some user input parameters which are context-dependent, such as the user's location and current local time. This type of parameters can be pre-filled based on user's contextual information. The value of a context-aware parameter can be obtained from two types of data sources:
  - *Direct Data Sources.* The context data is directly available and accessible with little computation, such as entries in Google calender, time from mobile phone clock, "my" To-do lists from task manager and a friend's preferences on Facebook.

[13] https://www.bestbuyrewardzone.ca/

- *Analyzed Data Sources.* With the accumulation of user history, additional contextual data can be mined from history data such as user behaviors. For example, a user always books a round-trip business class flight ticket from Toronto to Los Angeles for business trips and a round-trip economic class flight ticket from Toronto to Vancouver for personal trips.

In this paper, we consider only the user input parameters whose values can be obtained directly from available sources.

4.2 Second Empirical Study

The goals of the second empirical study are to: 1) examine the representativeness of the proposed categories of parameters; 2) validate the effectiveness of the approach of updating ontology; 3) test the effectiveness of the proposed approach for category identification; 4) validate the effectiveness of our proposed categories on improving existing approaches of pre-filling values to web forms.

4.2.1 Data Collection and Processing. We conduct our second empirical study on 45 websites from three different domains: Finance, Healthcare and Sports (*i.e.*, 15 websites from each domain) and 100 Android mobile Applications from five different categories: Cards& Casino, Personalization, Photography, Social and Weather in Google Play Market (*i.e.*, 20 applications from each studied category). The studied websites and mobile applications are different from the ones studied in our first empirical study. We use the same approach as discussed in section 4.1.1 to collect user input parameters from web and mobile applications. In total, there are 146 and 68 unique user input parameters from web and mobile applications respectively.

4.2.2 Research Questions. We presents four research questions. For each research question, we discuss the motivation of the research question, analysis approach and the corresponding findings.

RQ1. Are the proposed categories of parameters sufficient to cover the different types of user input parameters?

Motivation. The existing tools and approaches do not distinguish the characteristics of user input parameters. The parameters are processed equally. Therefore the lack of knowledge on the user input parameters negatively impacts the accuracy of pre-filling approaches, and is the main reason for limited support of reusing parameters within one application or across multiple applications. Categorizing the users input parameters can help us in understanding the different characteristics of parameters. Each category of parameters has its unique characteristics which need to be fully understood.

In Section 4.1, we identified 4 categories for user input parameters via a manual empirical study. In this research question, we validate the proposed categories to see if they can explain the further unseen user input parameters collected in the empirical study.

Analysis Approach. To answer this question, we study the user input parameters collected from the 45 websites and 100 mobile apps. For each user input parameter, we manually process it to see whether a user input parameter

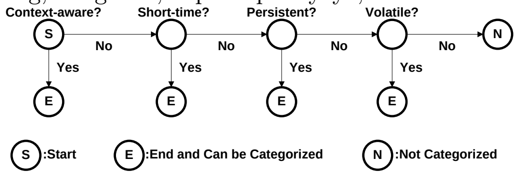


Fig. 7: The decision tree for the judgment process

belongs to any category or not. The judgment process is based on a decision tree as shown in Figure 7. When we process a parameter, we first decide whether it is a context-aware parameter or not. If no, we further decide whether the parameter can be categorized as a short-time parameter or not. If no, then we decide whether the parameter is a persistent parameter or not. If no, finally we decide whether the parameter is a volatile parameter or not. If it is still a no, the parameter cannot be categorized by using our proposed categories of input parameters.

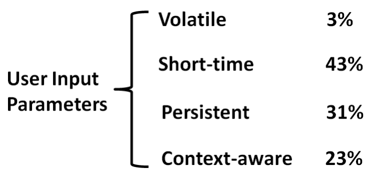


Fig. 8: The percentage of each category

Results. Figure 8 shows that all of the unseen user inputs parameters can be categorized into our proposed categories and there is no "not Categorized". More specifically, 43% of the parameters are short-time and only 3% of the parameters are volatile parameters. The results suggest that our categories are enough to describe all the unique user input parameters in our empirical study.

RQ2. Is the proposed approach for updating ontology effective?

Motivation. Usually the ontologies are created manually by web filling tool builders or contributors from the communities of knowledge sharing. It is helpful to have an automatic approach to update and expand the ontologies. We test the effectiveness of the WordNet-based approach of updating ontology proposed in Section 3.2.

Analysis Approach. We conduct two experiments to answer this question. In the first experiment, our approach for updating ontology uses the input parameters from one domain of applications to update the ontology to include the parameters from other domains of applications. In the second experiment, our approach for updating ontology uses the parameters from one domain of applications to update the ontology to include the parameters from the same domain of applications.

To conduct *experiment 1*, we construct an ontology[14], denoted as $O^{Initial}$, using the 76 user inputs parameters from the first empirical study (Section 4.1). Second, we use the ontology $O^{Initial}$ as the existing ontology to include the parameters of 146 parameters from web applications and 68 parameters from mobile applications (Section 4.2.1). Third, we compute the precision and the recall of

[14] https://dl.dropboxusercontent.com/u/42298856/icwe2014.html

the approach using Equation (1) and Equation (2) respectively. The precision metric measures the fraction of retrieved user input parameters that are placed in the correct place (*i.e.*, having the right relations with other parameters), while the recall value measures the fraction of correct user input parameters that are retrieved.

$$precision = \frac{|\{correct\,Parameters\} \bigcap \{retrieved\,Parameters\}|}{|\{retrieved\,Parameters\}|} \tag{1}$$

$$recall = \frac{|\{correct\,Paras\} \bigcap \{retrieved\,Paras\}|}{|\{correct\,Paras\}|} \tag{2}$$

The goal of *experiment 2* is to see whether we can obtain a better result if we build a domain dependent ontology for each domain in our empirical study. To conduct *experiment 2*, we conduct the following steps on the 45 websites:

1. randomly select 5 of 15 Finance websites and construct the domain specific ontology for the user input parameters in Finance using the user input parameters from the selected websites.
2. use the domain dependent ontology of parameters in Finance as an existing ontology and perform a manual updating on the existing ontology to include the user input parameters from the rest of 10 Finance websites.
3. apply the automatic ontology updating approach on the parameters from the rest of 10 Finance websites to verify its performance.

We replicate the previous steps on 15 Health Care websites and 15 Sports websites. We conduct our analysis on mobile applications in the same way as we do on websites, the only difference is that we randomly select 5 mobile applications from each domain. We compute the precision and the recall of the approach using Equation (1) and Equation (2) respectively.

Table 1: Results of our approach for updating ontology using domain-specific ontology

Type	Domain	Precision(%)	Recall(%)
website	Finance	83	66
website	Health Care	78	69
website	Sports	95	65
mobile	Cards&Casino	92	74
mobile	Personalization	87	48
mobile	Photography	90	62
mobile	Social	85	42
mobile	Weather	97	85

Results. In the experiment 1, our approach for updating existing ontology obtains a precision of 74% and a recall of 42% on websites, and a precision of 82% and a recall of 30% on mobile applications. We further investigate our results to gain further insights regarding the low recalls. We found that 35 of 146 website user input parameters and 25 of 68 mobile applications user input parameters can be found in the ontology constructed using the input parameters collected during the initial empirical investigation. The low recalls indicate that it is hard to use one general ontology as the existing ontology to include the parameters from other domains automatically, although the precisions are considerably high.

In experiment 2, Table 1 shows that our approach for updating ontology obtain a precision of 85% and a recall of 67% on websites, and a precision of 90% and a recall of 62% on mobile applications. The results of our approach in **experiment 1** can be improved by using the domain-specific ontology as the existing ontology. On average, the approach can be improved by 11% in terms of precision and 25% in terms of recall on websites, and 8% in terms of precision and 32% in terms of recall on mobile applications.

RQ3. Is our approach of category identification effective?

Motivation. After showing the representativeness of our categories for input parameters, we propose an ontology-based approach to identify a category for an input parameter. In this section, we investigate the effectiveness of our approach for identifying a category for an input parameter.

Analysis Approach. To assess the performance of the proposed approach, we proceed as follows:

1. We use the domain-specific ontologies constructed in RQ2 as the training ontologies.
2. We categorize the parameters in ontologies by assigning a category to each parameter.
3. We locate the nearest neighbours for the input parameters which are not in the training ontologies.
4. We apply our approach on the parameters in Step 3

We compute the precision and the recall of the approaches using Equation (1) and Equation (2) respectively. The precision metric measures the fraction of retrieved user input parameter that are categorized correctly, while the recall value measures the fraction of correct user input parameters that are retrieved.

Table 2: Results of our approach of category identification

Type	Domain	Precision(%)	Recall(%)
website	Finance	90	79
website	Health Care	92	82
website	Sports	89	70
mobile	Cards&Casino	90	78
mobile	Personalization	91	58
mobile	Photography	95	62
mobile	Social	77	55
mobile	Weather	100	88

Results. Table 2 shows the precision and recall values of our approach to identify a category for input parameters. On average, our approach can achieve a precision of 90% and a recall of 77% on websites, and a precision of 91% and a recall of 68% on mobile applications. Our approach works well on the input parameters from mobile applications in the domain of weather, because usually the weather mobile apps relatively have fewer number of input parameters and very similar functionalities.

RQ4. Can the proposed categories improve the performance of pre-filling?

Motivation. The existing pre-filling approaches (*e.g.*, [7][14]) treat all the input parameters equally. They do not take into consideration the characteristics of input parameters. These approaches are usually based on the string matching or semantic similarity calculation between the user inputs and the labels of input parameters (*e.g.*, input fields). Essentially they fill a value to an input field as long as a match between a user input and an input field is identified, however the value required by the input parameter may not be suitable for pre-filling due to curtain conditions as mentioned in Section 4.1.2. In this research question, we evaluate the effectiveness of our categories on improving the existing techniques of reusing user inputs.

Analysis Approach. To answer this question, we build a baseline approach which adopts the approach in [7]. The baseline approach [7] pre-fills values to input parameters based on the semantic similarity between user inputs and textual information (*e.g.*, words mined from labels and the attributes of HTML DOM elements defining the input parameters in the user interface) of input parameters. The baseline approach calculates the similarity between user inputs and labels of input parameters using WordNet [9]. The stopword removal, word stemming and non-English words removal are conducted on the textual information of input parameters to identify meaningful words. Then, WordNet is used to expand each word with synsets (*i.e.*, a set of words or collation synonyms) to retrieve synonyms of the found terms.

We build our approach by enriching the baseline approach with the proposed categories for input parameters (Baseline + Categories). We evaluate the improvement of applying our categories with the baseline approach. Our approach use the ontology-based approach proposed in Section 3.3 to identify a category for an input parameter to see whether the input parameter is suitable for pre-filling or not. If the input parameter is suitable for being pre-filled, our approach pre-fills an input parameter. We compute precision using Equation (1) and recall using Equation (2) to measure the improvement. The precision metric measures the fraction of retrieved user input parameters that are pre-filled correctly, while the recall value measures the fraction of correct user input parameters that are retrieved.

To pre-fill input parameters using the user previous inputs, we collect user inputs through our input collector [6] which modifies the Sahi[15] tool to track the user's Web activities. The first author of this paper used the tool to track his inputs on web forms from three domains: Finance, Health and Sports. We apply both the baseline approach and our approach on the 45 websites.

Results. Table 3 shows that our approach incorporating categories of input parameters can improve the baseline approach on average 19% in terms of precision. We further inspect the results and found that our approach can reduce

[15] http://sahi.co.in/

Table 3: Results of the evaluations of our categories on improving the baseline approach of filling input parameters.

Domain	Baseline		Baseline+Categories	
	Precision(%)	Recall(%)	Precision(%)	Recall(%)
Finance	50	34	64	36
Health	41	22	67	30
Sports	36	16	53	20

the number of wrong filled values compared with the baseline approach. Some of the wrong filled values should not be pre-filled to the input parameters in the fist place.

5 Threats to Validity

This section discusses the threats to validity of our study following the guidelines for case study research [10].

Construct validity threats concern the relation between theory and observation. In this study, the construct validity threats are mainly from the human judgment in categorizing the parameters and ontology construction. We set guidelines before we conduct manual study and we paid attention not to violate any guidelines to avoid the big fluctuation of results with the change of the experiment conductor.

Reliability validity threats concern the possibility of replicating this study. We attempt to provide all the necessary details to replicate our study. The websites and mobile apps we used are publicly accessible[16].

6 Related Work

In this section, we summarize the related work on form auto-filling approaches.

Several industrial tools have been developed to help users fill in the Web forms. Web browsing softwares provide Web form Auto-filling tools (*e.g.*, Firefox form auto-filling addon [2] and Chrome form auto-filling tool [3]) to help users fill in forms. RoboForm [11] is specialized in password management and provides form auto-filling function. Lastpass [12] is an on-line password manager and form filler. 1Password [13] is a password manager integrating directly into web browsers to automatically log the user into websites and fill in forms. These three tools store user's information in central space, and automatically fills in the fields with the saved credentials once the user revisit the page.

Some studies (*e.g.*,Winckler et al. [14] Bownik et al. [15] Wang et al. [16]) explore the use of semantic web for developing data binding schemas. The data binding schemas are essential techniques helping connect user interface elements with data objects of applications. This technology needs an ontology to perform the data integration. Instead of focusing on custom ontology, some binding schemas rely on the emergence of open standard data types, such as Microformats [17] and Micodata [18]. Winckler et al. [14] explore the effectiveness of the

[16] https://dl.dropboxusercontent.com/u/42298856/icwe2014.html

data schemas and the interaction techniques supporting the data exchange between personal information and web forms. Wang et al. [6] propose an intelligent framework to identify similar user input fields among different web applications by clustering them into different semantic groups. Araujo et. al [7] propose a concept-based approach, using the semantic of concepts behind the web form fields, for automatically filling out web forms. Some studies (*e.g.*, [19]) require apriori ([20]) tagging of websites, or a manually crafted list that includes the labels or names of input element to describe a semantic concept. These approaches can only be applicable to a specific domain or need explicit advice from the user. Hartman and Muhlhauser [4] present a novel mapping process for matching contextual data with UI element. Their method can deal with dynamic contextual information like calendar entries.

All the above approaches do not identify the variety of input parameters and treat input parameters equally by attempting to fill them into input parameters. They do not take into consideration the meaning of input parameters. In this study, we study the nature of input parameters and try to understand them from the end-user's point of view, not just by the similarity between the user inputs and the input parameters. Our study shows that taking into consideration that characteristics of input parameters via the identified categories significantly improves the performance of pre-filling.

7 Conclusion and Future Work

Reusing user inputs efficiently and accurately is critical to save end-users from repetitive data entry tasks. In this paper, we study the distinct characteristics of user input parameters through an empirical investigation. We propose four categories, *Volatile, Short-time, Persistent and Context-aware*, for input parameters. The proposed categories help pre-filling tool builders understand that which type of parameters should be pre-filled and which ones should not.

In this paper, we propose an ontology model to express the common parameters and the relations among them. In addition, we propose a WordNet-based approach to update the ontology to include the unseen parameters automatically. We also propose an approach to categorize user input parameters automatically. Our approach for category identification uses the proposed ontology model to carry the category information.

Through an empirical study, the results show that our proposed categories are effective to explain the unseen parameters. Our approach for updating ontology obtains a precision of 88% and a recall of 64% on updating the ontology to include unseen parameters on average. On average, our approach of category identification achieves a precision of 90.5% and a recall of 72.5% on the identification of a category for parameters on average. Moreover, the empirical results show that our categories can improve the precision of a baseline approach which does not distinguish different characteristics of parameters by 19%. Our proposed categories of input parameters can be the guidelines for pre-filling tool builders and our ontologies can be consumed by existing tools.

In the future, we plan to expand our definition of categories for input parameters by considering the scenarios of pre-filling web forms by multiple end-users. We plan to recruit end-users to conduct user case study on our approach.

Acknowledgments

We would like to thank all the IBM researchers at IBM Toronto Laboratory CAS research for their valuable feedback to this research. This research is partially supported by IBM Canada Centers for Advance Studies.

References

1. E. Rukzio, C. Noda, A De Luca, J. Hamard, and F. Coskun. Automatic form filling on mobile devices. Pervasive Mobile Computing, vol. 4, no. 2, pp. 161-181, 2008.
2. Mozilla Firefox Add-on Autofill Forms, `https://addons.mozilla.org/en-US/firefox/addon/autofill-forms/?src=ss`. Last Accessed on Feb 4th, 2014.
3. Google Chrome Autofill forms, `https://support.google.com/chrome/answer/142893?hl=en`, Last Accesed on Feb 4th, 2014.
4. Melanie Hartman and Max Muhlhauser. Context-Aware Form Filling for Web Applications. ICSC' 09. IEEE International Conference on Semantic Computing, 2009, pp. 221-228.
5. G. Toda, E. Cortez, A. Silva, E. Moura. A Probabilistic Approach for Automatically Filling Form-Based Web Interfaces. The 37th International Conference on Very Large Data Base, August 29th - September 3rd 2011, Seattle, Washington.
6. S. Wang, Y.Zou, B. Upadhyaya, and J.Ng. An Intelligent Framework for Auto-filling Web Forms from Different Web Applications. 1st International Workshop on Personalized Web Tasking, co-located with IEEE 20th ICWS, June 27, 2013, Santa Clara Marriott, California, USA.
7. S. Araujo, Q. Gao, E. Leonardi, J. Houben. Carbon: domain-independent automatic web form filling. In Proceedings of the 10th International Conference on Web Engineering (ICWE'10), Berlin, Heidelberg, 292-306.
8. H. Xiao, Y. Zou, R. Tang, J. Ng, L. Nigul. An Automatic Approach for Ontology-Driven Service Composition. Proc. IEEE International Conference on Service-Oriented Computing and Applications, Taipei, Taiwan, 14-15 December 2009.
9. WordNet: `http://wordnet.princeton.edu/`. Last Accessed on Mar 25th, 2013.
10. R.K.Yin, Case Study Research: Design and Methods-Third Edition, 3rd. SAGE Publications, 2002.
11. RoboForm: `http://www.roboform.com/`. Last Accessed on Mar 25th, 2013.
12. LastPass: `http://www.lastpass.com/`. Last Accessed on Mar 25th, 2013.
13. 1Password: `https://agilebits.com/`. Last Accessed on Mar 25th, 2013.
14. M. Winckler, V. Gaits, D. Vo, S. Firmenich, G. Rossi. An Approach and Tool Support for Assisting Users to Fill-in Web Forms with Personal Information, in SIGDOC' 11, Proceedings of the 29th ACM international conference on Design of communication, pp. 195-202, October 3-5, 2011.
15. L. Bownik, W. Gorka, A. Piasecki. Assisted Form Filling. Engineering the Computer Science and IT. InTech, October 2009. ISBN 978-953-307-012-4.
16. Y. Wang, T. Peng, W. Zuo, R. Li. Automatic Filling Forms of Deep Web Entries Based on Ontology. In Proceedings of the 2009 International Conference on Web Information Systems and Mining (WISM'09). Washington, DC, USA, 376-380.
17. R. Khare. Microformats: The Next (Small) Thing on the Semantic Web?. IEEE Internet Computing, vol. 10, no. 1, pp. 68-75, January/February, 2006.
18. I. Hickson. HTML Microdata. `http://www.w3.org/TR/microdata`. Last Accessed on March 25th, 2013.
19. J. Stylos, B. A. Myers, and A. Faulring. Citrine: providing intelligent copy-and-paste. in Proceedings of UIST, 2004, pp 185-188.
20. Apriori: `http://en.wikipedia.org/wiki/Apriori_algorithm`. Last Accessed on Mar 25th, 2013.