Automated Extraction of Personal Knowledge from Smartphone Push Notifications

Yuanchun Li, Ziyue Yang, Yao Guo, Xiangqun Chen
Key Laboratory of High Confidence Software Technologies (Ministry of Education)
Peking University
Beijing, China
Email: {liyuanchun, ziyue.yang, yaoguo, cherry}@pku.edu.cn

Yuvraj Agarwal, Jason I. Hong
School of Computer Science
Carnegie Mellon University
Pittsburgh, United States
Email: {yuvraj, jasonh}@cs.cmu.edu

***Abstract*—Personalized services are in need of a rich and powerful personal knowledge base, i.e. a knowledge base containing information about the user. This paper proposes an approach to extracting personal knowledge from smartphone push notifications, which are used by mobile systems and apps to inform users of a rich range of information. Our solution is based on the insight that most notifications are formatted using templates, while knowledge entities can be usually found within the parameters to the templates. As defining all the notification templates and their semantic rules are impractical due to the huge number of notification templates used by potentially millions of apps, we propose an automated approach for personal knowledge extraction from push notifications. We first discover notification templates through pattern mining, then use machine learning to understand the template semantics. Based on the templates and their semantics, we are able to translate notification text into knowledge facts automatically. Users' privacy is preserved as we only need to upload the templates to the server for model training, which do not contain any personal information. According to experiments with about 120 million push notifications from 100,000 smartphone users, our system is able to extract personal knowledge accurately and efficiently.**

***Index Terms*—Personal data; knowledge base; knowledge extraction; push notifications; privacy**

I. Introduction

Push notifications are widely used on mobile devices such as iPhone or Android smartphones. A push notification is a message that pops up on a mobile device and can be used for multiple purposes, such as SMS or social networking updates (e.g. your friend Alice sent you a message), travel schedule changes (e.g. your flight to Beijing is canceled), and shopping order delivery messages (e.g. the clothes you purchased has been shipped), just to name a few. Each user receives about 63.5 notifications per day on his/her smartphone[1].

This paper proposes an approach to extracting personal knowledge (`<user, relation, entity>` triples) from smartphone push notifications. Personal knowledge is a structured form of data that contains information about users' profile, behaviors, interests, etc. Such knowledge is important and useful for various mobile applications and mobile services, such as recommender systems [2], [3], virtual personal assistants [4], and authentication systems [5]. Researchers have also proposed methods for extracting personal knowledge from various other kinds of data sources such as user utterances [6] and communication logs [7]. In fact, many companies, especially smartphone vendors, have already started to make use of the personal knowledge extracted from different sources on smartphones to provide better services (e.g., emails, SMS messages and calendars). For example, Google[1] extracts and summarizes flight and hotel reservation information from emails with markup [8]. Apple Siri[2] reads users' calendar events to answer questions like "when is my next appointment". However, these approaches only deal with specific categories of personal knowledge, where the information is well-structured or programmatically available. Compared to them, push notifications are a more natural source of personal knowledge as they act like a proxy to many other data sources.

Extracting personal knowledge in general on smartphones is a difficult task. On one hand, there exists abundant personal information on smartphones, which can be exploited for many apps to provide better services to end users. On the other hand, it is not desirable to obtain full permissions to access personal information directly as protecting user privacy has also become a first-order priority for mobile apps [9].

In contrast, using push notifications as sources for personal knowledge offers several key benefits. First, push notifications contain and summarize a rich range of important personal information, such as user profiles, social relationships and information on everyday life. Second, push notifications are well-structured, as most of them are generated automatically using fixed templates, thus simplifying the task of extracting useful information from them. Third, notifications offer a uniform way of accessing data siloed across many apps.

Similar to the general knowledge base population (KBP), extracting personal knowledge from push notifications can be viewed as a slot filling task [10]. The entities related to the user (e.g. the Twitter accounts that the user follows, the products that the user purchases, etc.) are reserved as slots, and the goal of personal knowledge extraction is to collect entity values (slot fillers) from the large-scale push notifications. The state-of-the-art approaches [11], [12] for slot filling model the problem as a sequence labeling task and use RNNs to find both the boundaries and labels of slot fillers. However,

[1]https://developers.google.com/gmail/markup/google-now
[2]https://www.apple.com/ios/siri/

the entity values of personal knowledge in push notifications are often arbitrary phrases (e.g. @realDonaldTrump, iPhone X 64G Silver, etc.), making it extremely hard to find the entity boundary through sequence labeling.

A more straightforward solution is to define a template for each kind of push notifications manually, one that captures the semantics embodied in the notification. Once we have these templates, it becomes very easy to identify the relevant notifications and extract the related components to form a database of personal knowledge about the user. For example, here is a typical notification template: "Dear `$param1`, here are some `$param2` job opportunities for you". Besides the structure, we can easily understand that `$param1` is the name of the user, `$param2` is the user's profession, and the user is hunting for jobs. Applying this rule to a specific push notification "Dear David, here are some software engineer job opportunities for you", we are able to extract the parameters (`David` and `software engineer`) and generate knowledge triples: `<user, name, David>`, `<user, profession, software engineer>`, and `<user, status, job_hunting>`.

However, the above mentioned solution require defining the patterns or templates manually, and as such does not scale well, especially as the number of apps increases. Each of these apps might use a different template to construct their notifications. To solve this challenge, this paper proposes an automated approach to identify notification templates and to learn their semantics. Specifically, our proposed approach includes the following steps: it first discovers the notification templates on the device, then uploads the templates to the server for offline learning to train a model to understand the semantic meaning of each template, and finally, it is able to extract personal knowledge based on the server-trained model.

We achieve the following goals with the proposed approach: (1) we are able to automatically identify the templates for different types of notifications, including formerly unseen new templates; (2) we can also understand the meaning of each new template through an offline learning phase; (3) because we only need the templates (without specific user information) to train the model, we do not need to send sensitive user information out of the devices, thus helping preserve user privacy during the whole process.

To evaluate our approach, we conducted experiments on around 120 million real notifications from 100,000 smartphone users. The results show that our system is able to discover notification templates with a precision of 86.8% and understand the semantics of unseen templates accurately (around 83% F1-score for templates of new apps and 91% F1-score for new templates of existing apps). We also demonstrate that the discovered templates and the semantic model can be directly used to extract personal knowledge from push notifications.

This paper makes the following main contributions:

- To the best of our knowledge, this is the first work to propose that push notifications can be used as a data source for personal knowledge extraction on mobile devices such as smartphones. We also introduce an automated and privacy-preserving method to extract personal knowledge from push notifications.
- We implement a prototype system for personal knowledge extraction on Android. The system can run on smartphones to support personalized services such as recommender systems and virtual personal assistants.
- We evaluate our approach on around 120 million real-world push notifications from 100,000 users. The results show that our method is able to extract personal knowledge with a high accuracy.

II. Background

A. Push Notifications

Push notifications serve as a core feature for mobile devices such as smartphones and tablets. They are mainly used by the operating system and smartphone apps to inform users of various of events, such as the availability of a software update, the arrival of a message, the status update of an online purchase, the recommendation of news and articles, etc. As notifications can be displayed without activating the apps' normal UI, they are a preferable way used by app developers to deliver information to users promptly. As a result, push notifications sometimes contain valuable information.

The content of smartphone push notifications can be generated either locally or remotely. Most operating systems provide APIs for apps to display notifications locally. For example, Android allows apps to define a `Notification` instance, set a title and a text body, and send it as an `Intent` to display. Most notifications of system events such as alarms and device status updates are generated with this method. Many systems and third-party services also provide a way for developers to construct notifications on the server, then push them to the client devices. Remotely-generated push notifications are more dynamic and less structured as compared to local notifications, since any online-service notifications such as messages, news, and advertisements can be generated remotely.

Most push notifications are automatically generated with templates. However, because each push notification typically contains personalized content, an app can customize the template parameter for each user to achieve this goal, while the templates remain the same across different users. Thus, it is possible to extract the personal information from push notifications once we know the templates.

B. Knowledge representation

Existing knowledge bases such as YAGO [13], Freebase [14] and Google Knowledge Graph [15] use relational knowledge representations. Information is modeled in the form of entities and relations between them. Such kind of representation has been widely used in the area of logic and artificial intelligence [16].

The W3C Resource Description Framework (RDF) [17] defines an abstract syntax for relational information representation. The core structure of the abstract syntax is a set of *subject, predicate, object* (SPO) triples, where subjects and objects are entities, while predicates are defined as the

TABLE I: The ontology of the personal knowledge considered in this paper. We considered 11 types of knowledge relations (column 2) in 4 categories (column 1). For each relation, we show several examples of entities (column 3) and a sample notification text (column 4). Note that the examples are simplified (and translated) for better presentation.

Category	Relation	Example entities	Example notification
User profile	name	Alice, Bob1997, ...	Hi Alice, here are some recommended reads for you.
	gender	male, female, ...	Dear Mr. Li, please review your receipt.
	profession	doctor, software engineering, ...	7 software engineer positions for you: ...
	status	in_college, job_hunt, ...	Facebook: found 9 classmates in Stanford University.
Social	follows	Justin Bieber, @realDonaldTrump, etc.	Justin Bieber posted a new photo.
	isFriendOf	Candy's Mother, David, etc.	David sent you a message: ...
Location	livesNear	Beijing, MIT campus, etc.	Beijing weather today: 6 C, sunny.
	travelsTo	Sweden, Tokyo, etc.	Flight CU1234 from Beijing to Tokyo is going to take off.
Shopping	purchases	iPhone X 64G Silver, milk powder, etc.	Your order iPhone X 64G Silver has been shipped.
	wantsToBuy	NIKE Men's Roshe Run Size 10, beer, etc.	The beers in your shopping cart is on sale.
	visitsMerchant	Walmart, Wendy's, etc.	Thank you for shopping at Walmart.

relations between them. All existing knowledge bases can be represented with such SPO triples.

Similar to the world's general knowledge bases, the facts in personal knowledge bases can also be represented as SPO triples. Li *et al.* [6] represent personal knowledge as a user-centered graph, in which the subjects of all knowledge triples are the user. They follow the Freebase semantic knowledge graph schema, including 18 types of <user, relation, entity> triples, such as <user, place_of_birth, New York City>, <user, parents, Rosa>, etc. We follow the definition of Li *et al.*, but we use a different set of relations that frequently appear in push notifications.

The ontology of the personal knowledge considered in this paper is shown in Table I. we consider four common categories of personal knowledge, including user profile, social relationship, location and shopping. Other categories of knowledge can be easily added in the future.

III. Personal Knowledge Extraction

We propose an automated approach to extract personal knowledge facts from push notifications on smartphones. The problem is defined as follows. Suppose there are a set of smartphone users, each with a list of push notifications. Our goal is to extract knowledge triples (<user, relation, entity>) from the notification content for each user, with little-to-none manual efforts.

Of course, developers can always define templates manually. However, because there are too many notifications, it takes a lot of efforts to manually identify and define all the different templates for all apps. Moreover, there are always new templates, which cannot be covered by existing templates defined manually. In contrast, we expect that an automated approach can identify new notification templates automatically, as well as the meaning for each notification.

Our approach is mainly based on the observation that knowledge is formatted into notifications with templates. The templates can then help identify knowledge entities and understand their meanings as well. We aim to identify the notification templates automatically, and then understand their structure and meanings through machine learning.

A. Approach Overview

Figure 1 shows an overview of our proposed approach in extracting personal knowledge from push notifications. The approach consists of three phases: template learning, template understanding, and knowledge extraction. The template learning phase runs on the server and the other two phases run on users' devices. The main purpose of template learning is to train a machine learning model to understand the semantics of notification templates. Then in the template understanding phase, the trained model can be used to infer what types of personal knowledge triples that each notification template may express. The discovered templates and the inferred semantics are then used to identify template parameters (i.e. entities) from notification text and generate personal knowledge triples.

Consider an example notification with order shipping information: "Your order iPhone X has been shipped". For each type of notification, we assume that there are other similar notifications on the device, such as "Your order Nike Running Shoes has been shipped". By mining patterns from all notifications, we can discover the template for this notification: "Your order $param has been shipped", where $param is a parameter to the template. The template parameters are potentially personal knowledge entities, as they are usually customized for different users.

However, discovering the template of a notification is only the first step. Once we extract the relevant elements from a notification, we still need to understand the meaning of each component from the notification. In order to infer whether the template is a personal knowledge template and what knowledge triples the template may have, we use a server-trained semantic model to understand the template. The semantic analysis is modeled as a multi-label classification problem: given a notification template, predict what kinds of personal knowledge triples it may express. Specifically, given the template "Your order $param has been shipped", the semantic model will predict a purchases relation for param, which leads to a knowledge triple templates <user, purchases, $param>. The model will also try to predict no-parameter knowledge triples (such as <user, gender, male>) based on the whole template.

The mapping from the notification template to the knowl-

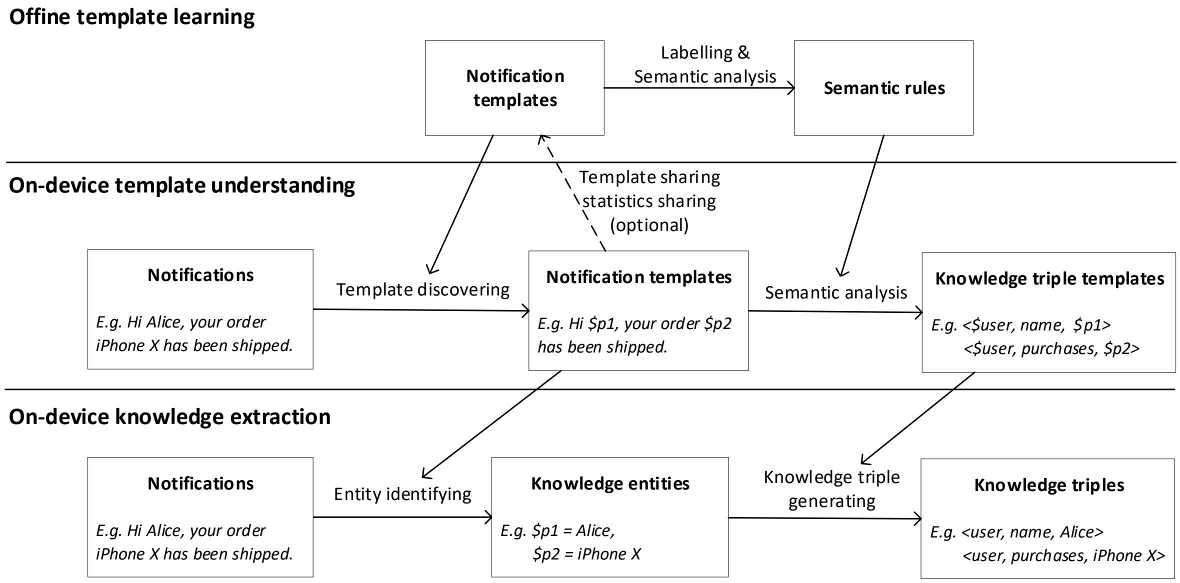


Fig. 1: An overview of our approach. On each user's device, we first identify templates from the notifications, then infer template semantic rules using a server-trained semantic model. The templates and semantic rules are used to identify personal knowledge entities and generate knowledge triples.

edge triple templates is referred to as a *template semantic rule* in this paper. By applying the template semantic rule to the original notification content, we are able to extract the knowledge triples: `<user, purchases, iPhone X>`.

The whole process introduces two main challenges: discovering notification templates and understanding template semantics. The following sections describe how we solve these problems through pattern mining and machine learning, respectively.

B. Template Discovering

The purpose of template discovering is to recover the templates that are used by app developers to generate notifications. There are two main challenges in discovering the templates. The first is that templates vary quite a bit in terms of personal data used, across different apps, and across different app versions. Altogether, these differences make it impractical, if not impossible, to manually summarize a complete list of templates. The second challenge is that the notifications on users' smartphones are usually privacy-sensitive. Thus uploading all notifications to a server for joint analysis is undesirable because it may cause the leakage of personal information.

In our solution, the template discovering process runs locally on each user's smartphone. It aims to identify templates that have at least two instances (i.e. two notifications generated from the same template) on a user device. The whole process involves several steps including notification filtering, notification clustering, and template extracting, as illustrated in Figure 2.

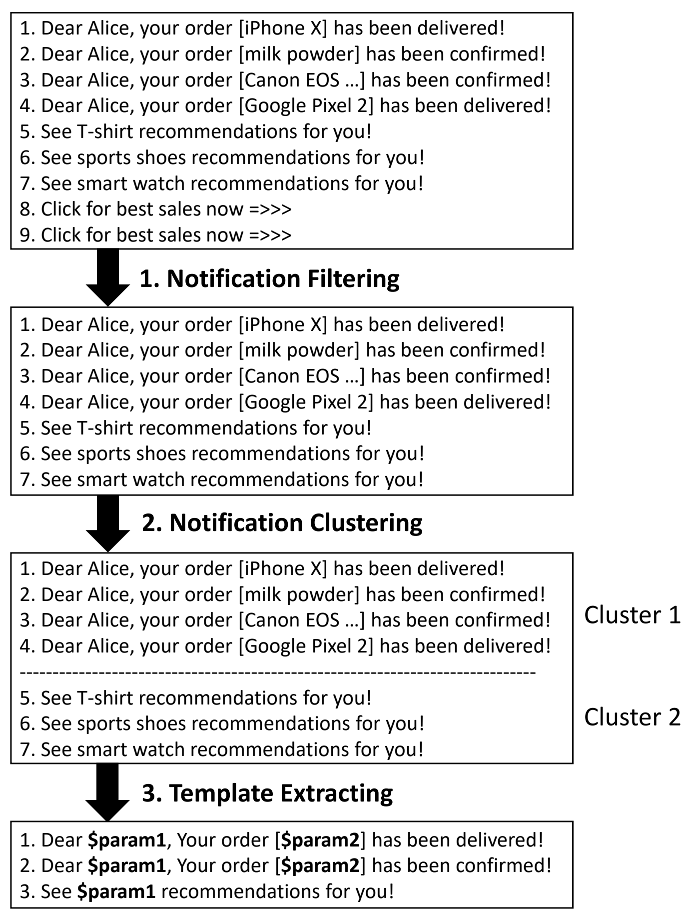


Fig. 2: An illustration of the template discovering process

1) Notification Filtering: We first preprocess the notifications by filtering out duplicated and unstructured ones. Duplicated notifications are typically system events or advertisements that usually do not contain personal knowledge.

Unstructured notifications, i.e. notifications generated without a template, are mainly messages or emails from other users. We identify and remove such notifications with several heuristics rules, for example checking the host app against a list of messenger apps and/or matching the notification text to known patterns, such as "[NEW MAIL](.+)".

2) Notification Clustering: After filtering, most of the remaining notifications are generated with templates. Given the fact that a template is a common subsequence of the notifications generated with it, extracting templates from notifications is similar to the task of longest common subsequence (LCS) mining. However, mining LCS from these notifications can still be hard, as the notifications may be significantly different from each other. A common solution, as used by Fu *et al.* [18] for log analysis, is clustering the items before mining patterns from each cluster. Inspired by their work, we first cluster the notifications before extracting templates from them.

We choose DBSCAN [19] as the clustering algorithm as the exact number of clusters is unknown. The distance metric we used in DBSCAN is the edit distance, where each edit operation can be adding, deleting, or replacing one word. This is intuitive as the notifications are originally generated from templates by simple editing (adding entity values as parameters).

As shown in Figure 2, each group of notifications after clustering are generated with one template or several very similar templates. This step also filters out a good deal of noisy data, i.e. notifications not belonging to any cluster.

3) Template Extracting: To extract notification templates, we first mine longest common subsequences (LCS) in each cluster of notifications. For example, the longest common subsequence extracted from cluster 1 in Figure 2 is "Dear Alice, your order `$param1` has been `$param2`". "iPhone X" and "milk powder" are possible values of `$param1`, while "delivered" and "confirmed" are possible values of `$param2`.

Unfortunately, the LCS cannot be directly used as notification template. First, the user name "Alice", which should be a parameter in the template, is not correctly identified. Second, "delivered" and "confirmed" are identified as parameter values, but they are not personal entities and should be a part of template. The two mistakes are both due to parameter misidentification, i.e. parameters misidentified as template text or template text misidentified as parameter.

We introduce *global word frequency* to address the problem. The global frequency of a word is the number of users having at least one notification containing this word. The parameter values such as "Alice" and "iPhone" are usually user-specific, thus should have low global frequencies, while template words such as "delivered" and "confirmed" should have high global frequencies as they are usually user-agnostic. Based on the global word frequencies, we are able to extract two templates from the cluster: "Dear `$param1`, your order `$param2` has been shipped" and "Dear `$param1`, your order `$param2` has been delivered". To guarantee user privacy, notification words are hashed before uploading to server to calculate global word frequencies.

Finally, the notification templates extracted on user devices are uploaded to the server to determine the final set of templates. As templates used by each app are typically the same for different users, our approach only requires a small portion of users to discover templates on their devices and share the templates. Other users can directly download and use the templates without running the template discovering phase. Meanwhile, sharing the templates should have little impact on privacy since the template itself does not contain any personal information.

TABLE II: Examples of template semantic rules. The first column shows the examples of notification templates. The second column lists the templates of knowledge triples extracted from the notification. `u`, `$p1` and `$p2` are short for "user", "parameter 1" and "parameter 2".

Notification template	Knowledge triple templates
Good news! `$p1` is on sale!	-
Your flight to `$p1` is delayed.	`<u, travelsTo, $p1>`
Hi `$p1`, your order `$p2` has been shipped.	`<u, name, $p1>` `<u, purchases, $p2>`
Mr. `$p1`, please review the receipt.	`<u, name, $p1>` `<u, gender, male>`
Here are some `$p1` job opportunities for you.	`<u, profession, $p1>` `<u, status, job_hunt>`

C. Template Semantic Rules

The discovered notification templates can be used to understand the sentence structure of the notifications. To extract personal knowledge, we will need to further understand the semantics of each template.

Knowledge extraction in this kind of scenarios is typically modeled as as a slot filling problem [10]: given a document and a slot to fill (i.e. a knowledge triple with a pending entity), finding the boundaries of the slot filler (i.e. identifying the entity value). The accuracy might be low if the document is poorly structured [10]. Our approach can easily understand the sentence structure with the help of notification templates. Thus the slot filling task is largely simplified: we do not need to determine the entity boundaries as they are automatically given by templates.

Due to the simplification, we are able to construct rules to extract knowledge from push notifications based on their templates. We introduce *template semantic rules* to help convert a notification to personal knowledge triples. A template semantic rule is defined as a mapping from a notification template to a list of *knowledge triple templates* (*KTTs* in short).

There are two types of *KTTs* considered in our approach, including *0-parameter KTTs* and *1-parameter KTTs*. An *1-parameter KTT* can be used to generate different knowledge triples based on what parameter is used for the entity value. For example, `<user, travelsTo, $param>` can use different location names (such as `New York City`, `China`, etc.) as the parameter. <user, purchases, $param> can use different product names (such as iPhone X, Nike Shoes, etc.) as the parameter. A *0-parameter KTT* is a template with a fixed entity value, which can only generate one type knowledge triple. *0-parameter KTTs* are suitable for attributive knowledge triples such as <user, gender, male>, <user, status, job_hunt>, etc.

Table II shows some examples of template semantic rules. For example, "Good news! $param1 is on sale" does not map to any personal knowledge triple. "Here are some $param1 job opportunities for you" is a personal knowledge template that maps to two knowledge triple templates (*KTTs*), including one *1-parameter KTT* (<user, profession, $param1>) and one *0-parameter KTT* (<user, status, job_hunt>). With the template semantic rules, we are able to extract personal knowledge triples by filling parameter values into the parameter slots of KTTs.

D. Automated Template Semantic Rule Generation

Although manually labeled semantic rules are the most accurate when used to understand notifications formatted with known templates, there might be a lot of unseen templates used by different or newer versions of apps. It is time-consuming and impractical to manually label semantic rules for all templates, as they may be generated by potentially millions of different apps. Thus, we extend our system to automatically generate semantic rules for unseen templates, based on a set of manually defined semantic rules for known templates.

We model the problem as a sequence classification problem: given a notification template, predict what knowledge triple templates (*KTTs*) it may represent.

We use an RNN-based method to address the problem. Figure 3 illustrates our model for automated semantic rule generation. For each notification template, we first represent each word in the template as a vector through word embedding. We use an existing word embedding model pre-trained with fastText [20], which is able to generate reasonable word embeddings for unseen words. Reusing the pre-trained model enables us make use of the meanings of words learned from large corpus, thus facilitates training our model with relative small dataset. The word vectors are then fed into a Bi-LSTM network, with which each word has a node capturing information from prefix words as well as a node capturing information from suffix words. By concatenating the output of the two nodes for each parameter, we can generate a vector representation of the parameter. Similarly, the whole template is represented as the concatenation of the outputs of the last word's forward node and the first word's backward node. Finally, the parameter vector of each parameter is used to predict 1-parameter KTTs, i.e. the knowledge triples that use the parameter as the entity value. The template vector is used to predict 0-parameter KTTs.

We use the manually labeled semantic rules (mappings from notification templates to knowledge triple templates) as to train the model. The trained model is used to predict knowledge triple templates for unseen templates. Both the automatically-predicted and manually-labeled semantic rules are used to extract knowledge triples from push notifications.

IV. Implementation and Evaluation

We implemented two versions of the proposed knowledge extraction mechanism for production and experiments respectively:

1) The production version is implemented as an Android app. The app runs as a background service, collecting received notifications, identifying the template for each notification, and extracting personal knowledge from it. The service also provides APIs for other apps to access the personal knowledge base.
2) The experiment version is implemented entirely on a server. It takes a centralized dataset with all user notifications as the input and extract knowledge triples for each user. However, we simulate the situation of the production version where the notifications are distributed on each user device, by processing each user data separately. This version of implementation is easier for conducting experiments as it does not require users to install our app.

We evaluated our proposed approach by primarily looking at two aspects:

1) Can our system discover new personal knowledge templates from smartphone notifications accurately?
2) Can our system understand the meaning of the templates accurately, especially previously unseen templates?

To answer these questions, we conducted experiments on a dataset of push notifications from real users.

A. Dataset Overview

The dataset we used in the experiments contains 119,289,901 notifications from 100,000 smartphone users, obtained through a mobile service provider in China. As these notifications may contain sensitive user information, all data have been collected from a group of designated test users in accordance with the policies of the service provider. We have strictly followed the "terms and conditions" specified by the smartphone provider with respect to these test users in our study. For example, the identities of all users have been anonymized, while all push notifications have been kept on the servers within the provider's company throughout the whole process.

The notifications were generated by 2,658 apps during 30 days from March to April 2018. Each notification entry is consisted of:

- A user ID: the unique identifier of the sampled smartphone. User IDs are anonymized for security reasons.
- An app ID: the unique identifier of the app that the notification belongs to.
- A timestamp: the time when the notification was pushed to the user's smartphone.

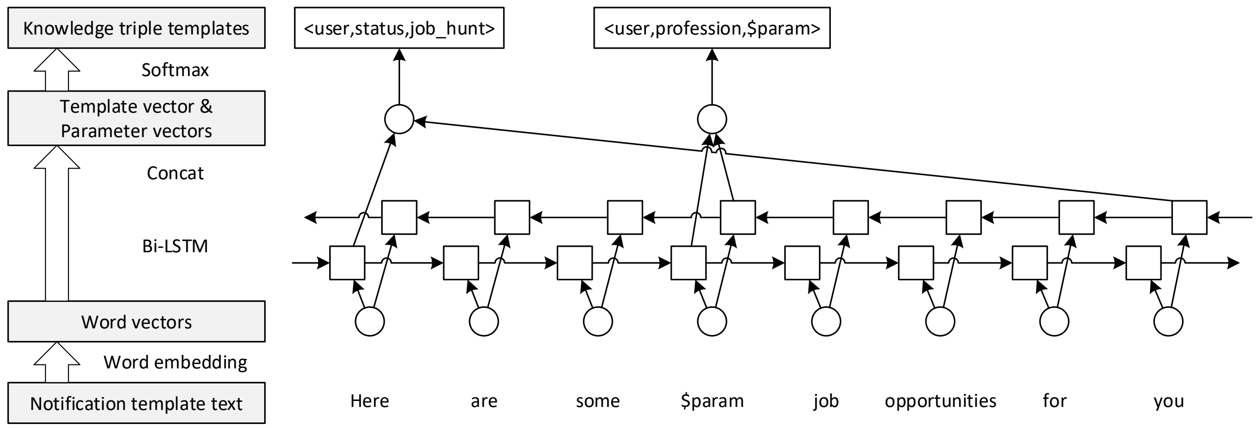
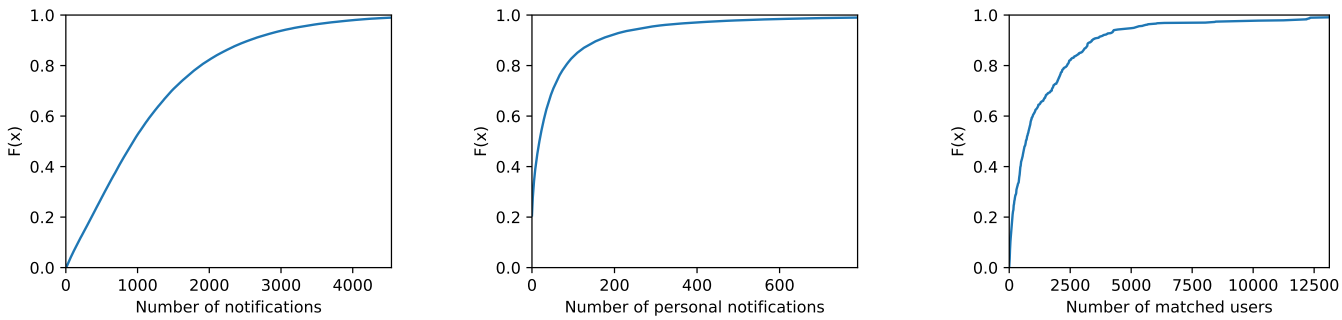


Fig. 3: The overview of our semantic rule prediction model.



(a) Number of notifications per user. (b) Number of personal notifications per user. (c) Number of matched users per template.

Fig. 4: Statistics of the dataset used for experiments

- Notification content: the notification title followed by the text body. All numbers in the notifications are elided for security and privacy reasons.

Figure 4a shows the distribution of the number of notifications per user in our dataset. Most users (around 50%) have more than 1000 notifications and about 20% of the users have more than 2000 notifications. On average, there are about 40 notifications for each user per day in our dataset, which is a subset of users' notifications. The reason is that we have only obtained the notifications that are pushed through a certain service provider, instead of all the notifications on a smartphone. However, while there is some bias in our data set, we believe that our technique can still generalize.

B. Accuracy of Notification Template Discovering

We simulated the scenario that the notifications are distributed on users' smartphones, and used the template discovering method described in Section III-B to identify notification templates.

In total, we discovered 2,788 templates belonging to 409 apps from the dataset. We manually labeled the discovered templates, determining whether each template contains personal knowledge and whether it is correct (by correct we mean that the template correctly identifies the boundaries of parameters). The result shows that there are 2,163 personal knowledge templates, among which 2,006 (92.7%) are correct, and 625 non-personal templates, among which 414 (66.2%) are correct. The overall correctness ratio of template discovering is 86.8%. The correctness for non-personal templates is relatively low because those notifications are usually less-structured. Such notifications include top news, sales information, etc., many of which are manually crafted without a template. However, it is not a huge problem as we are not going to extract knowledge from these non-personal templates anyway. Figure 4c shows the distribution of the number of matched users per personal knowledge template (i.e. the users who have one or more notifications matching the template). On average, each personal knowledge template has matched 1,571 users.

About 5.7% of all notifications contain personal information[3]. Specifically, 6,824,002 out of 119,289,901 notifications are identified as personal notifications, as they can match one of the discovered personal knowledge templates. The distribution of the number of personal notifications per user is shown in Figure 4b. Around 10% of the users have more than 200 personal notifications. On average, there were at least 68 push notifications for each user that contain personal knowledge. As we only considered a subset of notifications

[3]The proportion can be higher if we consider more types of personal knowledge beyond what have been defined in this paper.

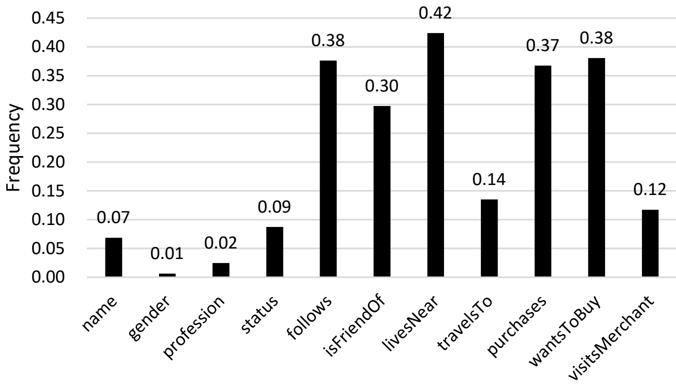


Fig. 5: Number of matched users per knowledge type.

due to the limitation of our dataset, we believe that more personal notifications can be found on actual smartphones.

C. Accuracy of Template Semantic Analysis

We also conducted experiments to evaluate how well our system can correctly understand the semantics of previously unseen templates. We used the 2,788 notification templates from 409 apps discovered with our template discovering method, as described in Section IV-B.

We manually labeled the knowledge triple templates for each notification template according to our personal knowledge ontology (Table I) as the ground truth. Figure 5 shows the number of users that have each type of knowledge (a user is identified to have a type of knowledge if one of his/her notifications matches a template in that knowledge type). As we can see, only a small portion of users have user-profile-related knowledge in their notifications, while lots of them can be extracted knowledge in social relationship, location, and shopping categories. This is because only few apps put user's personal information (name, gender, profession, etc.) into notifications and even fewer users are using these apps.

Each knowledge triple template is a label in our classification model, and each notification template can have zero or more labels. Labels are selected in order that there are enough samples (notification templates having the label) for machine learning. For example, the knowledge triple templates with `gender` relation and `profession` relation are not used as label in this experiment as they don't have enough samples. In total, we selected two `0-parameter KTTs` to evaluate the classification of templates and seven `1-parameter KTTs` to evaluate the classification of parameters, as shown in Table III.

We considered two situations where we need to predict semantic rules for unseen notification templates. The first is that when a new app is added to our system, all of the notification templates used by that app are unseen. The second is that when an existing app is updated, it may use a new template to bring information to its users. We designed two experiments to evaluate our system's performance for both situations:

- We first ran a 5-fold cross-validation to check whether our system is able to handle unseen templates from unseen apps. For each label, we randomly divided the 417 apps into 5 sets such that each set had approximately equal amount of apps whose templates have the label. For example, each set will have about 9 apps that contain `<u, follows, $p>` knowledge triples, as there are in total 45 apps containing such knowledge.
- In the second experiment, we also use 5-fold cross-validation, but with different partitioning method. For each app, we randomly divided its templates to 5 sets. We used 4 sets for training and the remaining 1 set for predicting in each fold.

The results of the both experiments are shown in Table III. Overall, our semantic model is able to accurately predict labels (i.e. generate semantic rules) for unseen templates. The accuracy of predicting labels for new templates of existing apps is high (89.41% precision and 92.19% recall), which is not a surprise because the new templates of an app are usually similar to its old templates.

For unseen apps' templates, the precision (83.62%) and the recall (82.56%) are both lower than the other situation. Is is because that different apps may use significantly different ways to express same types of knowledge. For example, a live streaming app (such as Twitch) may notify users about the updates of their subscribed anchors using "`$param` is live streaming.", while an online publishing app (such as Medium) may notify users about the updates of their favorite authors using "`$param` posted a new article.", both notification templates express a *following* social relationship while using totally different vocabularies. However, the accuracy is acceptable for most common use cases of personal knowledge, such as recommender systems and conversational bots.

Among the knowledge relations considered in our evaluation, the accuracy for "livesNear", "purchases" and "wantsToBuy" is relatively low. One major reason is that these relations can be expressed in a wide range of ways, while our dataset only contains a limited number of apps, each using a specific way to express the knowledge. We think this problem can be tempered by adding more training data, such as more labeled notification templates from other apps.

V. Limitations and Future Work

In this section, we highlight some of the limitations of our system and discuss possible solutions.

Strong privacy guarantee. Our system is privacy-preserving because it only uploads the notification templates to the server. However, it is not a strong privacy guarantee because the uploaded templates may contain personal information if the templates are incorrect. One possible solution is to scan potential templates for sensitive information before uploading, e.g. using Named Entity Recognition techniques.

Real-world scenario. Our system is evaluated with a dataset provided by a push notification service provider, which only contains remotely-generated notifications from a small subset of apps. The real scenario might be different as we will be

TABLE III: Accuracy of template semantic analysis. We used our model to predict labels (knowledge triple templates) for unseen notification templates, including the templates used by unseen apps and the new templates of existing apps.

KTT Category	KTT	#apps	#templates	Templates of unseen apps precision	Templates of unseen apps recall	New templates of existing apps precision	New templates of existing apps recall
0-param	<u, status, job_hunt>	8	48	92.31%	71.79%	94.13%	93.89%
	<u, status, car_hunt>	10	97	92.40%	77.37%	91.27%	89.68%
1-param	<u, name, $p>	22	90	87.81%	85.45%	93.77%	95.91%
	<u, follows, $p>	45	217	90.37%	76.21%	90.85%	93.89%
	<u, isFriendOf, $p>	129	485	87.12%	86.52%	90.21%	92.95%
	<u, livesNear, $p>	16	157	74.13%	71.40%	93.94%	87.58%
	<u, travelsTo, $p>	9	39	93.38%	86.72%	89.66%	92.10%
	<u, purchases, $p>	22	83	73.30%	86.85%	87.50%	91.79%
	<u, wantsToBuy, $p>	41	234	76.93%	84.45%	82.31%	90.88%
Overall		302	1450	84.50%	81.86%	89.69%	92.08%

able to access all notifications on users' smartphones. In the future, we would like to deploy our system and evaluate the performance of our system in the real-world scenario.

Comprehensive personal knowledge ontology. We considered three common categories of personal knowledge in our implementation. However, a lot of other knowledge categories can be found in push notifications, such as work information, travel information, etc. Meanwhile, our knowledge ontology is specifically designed for personal knowledge in push notifications. There ought to be a more complete and formal ontology of personal knowledge, like the one defined in Schema.org [8] for world's knowledge.

Other ways to obtain notification templates. Our approach requires a portion of users to upload discovered notification templates for offline learning. This requirement might be hard to fulfill if our system does not have enough users. We can solve this problem by using other methods to obtain an initial set of notification templates, such as extracting the templates from application code through static analysis, or generating notifications by automatically testing the apps [21].

VI. Related Work

A. Personal Knowledge Extraction

Personal knowledge extraction has attracted researchers' interests. The commonly used data sources of personal knowledge include conversational dialogs, SMS messages, sensor, and UI content. For example, Vhaduri *et al.* [22] proposed to mine users' places of interest and mobility patterns from mobile data. Min *et al.* [7] analyzed communication logs to infer users' social relationships. Spolaor *et al.* [23] presented a tool to extract UI interaction data for user habit analysis. Li *et al.* [6] introduced a statistical language understanding approach to construct personal knowledge graph from conversational dialogs.

As the data sources of personal knowledge are often privacy-sensitive, a lot of approaches have been proposed to mitigate the privacy concern during knowledge mining. PrivacyStreams [24] introduced an Android library for app developers to process personal data locally and transparently. PERUIM [25] analyzed the privacy sensitivity of apps' UI content. Appstract [26] used an offline user-agnostic learning phase and an on-device predicting phase to preserve the privacy of UI content analysis. Inspired by Appstract, our system also adopts a two-phase method to minimize users' privacy concern.

B. Notification Analysis

Prior to our work, push notifications have been studied by researchers from different aspects. Most researchers have focused on the disruptive nature of push notifications [27], [28], [1], [29] in order to guide the design implementation of better notification systems. For example, Shirazi *et al.* [27] analyzed a large scale of notifications and revealed the differences in the importance of notifications. Mehrotra *et al.* [28], [29] analyzed how notifications with different content, sender and context can cause disruption to users. Other researchers have also explored the effects of push notifications to foster meta-learning [30] and self-logging [31]. Our work does not aim to improve the notification mechanisms or use notifications to influence users' behaviors, instead, we use notification content for a broader purpose: building a personal knowledge base.

C. Wrapper Generation

Wrapper in data mining is a program that extracts content of a particular information source and translates it into a relational form. The aim of a wrapper is to locate relevant information in semi-structured data and to put it into a self-described representation for further processing [32]. Typically, wrappers are used to extract structured data, such as telephone directories, product catalogs, etc. from web pages formatted with fixed HTML templates. Wrappers can be generated manually by experts, semi-automatically through supervised learning [32], [33], [34], or automatically through unsupervised learning [35], [36], [18]. Our work deals with another form of wrapper: push notification templates. We used an unsupervised approach to discover the templates and supervised machine learning to understand the semantic rules of the templates.

VII. Concluding Remarks

This paper proposes an approach for extracting personal knowledge from smartphone push notifications. It is able to automatically identify templates from notification text using pattern mining techniques, and then understand the semantics of the templates through supervised machine learning. The approach is privacy-preserving as only the templates might be uploaded to the server for labeling and learning.

We have implemented a prototype system on Android and evaluate it with three common categories of personal knowledge. Experiments on about 120 million push notifications from 100,000 smartphone users show that we are able to discover and understand notification templates accurately, while successfully use the templates to extract personal knowledge from push notifications.

ACKNOWLEDGMENTS

This work was partly supported by the National Key Research and Development Program (No. 2017YFB1001904) and the National Natural Science Foundation of China (No. 61772042). Yao Guo (email: yaoguo@pku.edu.cn) is the corresponding author.

REFERENCES

[1] M. Pielot, K. Church, and R. de Oliveira, "An in-situ study of mobile phone notifications," in *Proceedings of the 16th International Conference on Human-Computer Interaction with Mobile Devices and Services*, ser. MobileHCI '14. New York, NY, USA: ACM, 2014, pp. 233–242.

[2] G. Iyer, D. Soberman, and J. M. Villas-Boas, "The targeting of advertising," *Marketing Science*, vol. 24, no. 3, pp. 461–476, 2005.

[3] Z. Wang, J. Liao, Q. Cao, H. Qi, and Z. Wang, "Friendbook: A semantic-based friend recommendation system for social networks," *IEEE Transactions on Mobile Computing*, vol. 14, no. 3, pp. 538–551, March 2015.

[4] T. R. Gruber, "Siri, a virtual personal assistantbringing intelligence to the interface," 2009.

[5] L. Guo, C. Zhang, J. Sun, and Y. Fang, "A privacy-preserving attribute-based authentication system for mobile health networks," *IEEE Transactions on Mobile Computing*, vol. 13, no. 9, pp. 1927–1941, Sept 2014.

[6] X. Li, G. Tur, D. Hakkani-Tur, and Q. Li, "Personal knowledge graph population from user utterances in conversational understanding," in *2014 IEEE Spoken Language Technology Workshop (SLT)*, 2014, pp. 224–229.

[7] J.-K. Min, J. Wiese, J. I. Hong, and J. Zimmerman, "Mining smartphone data to classify life-facets of social relationships," in *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, ser. CSCW '13. New York, NY, USA: ACM, 2013, pp. 285–294.

[8] R. V. Guha, D. Brickley, and S. Macbeth, "Schema. org: evolution of structured data on the web," *Communications of the ACM*, vol. 59, no. 2, pp. 44–51, 2016.

[9] J. Wiese, S. Das, J. I. Hong, and J. Zimmerman, "Evolving the ecosystem of personal behavioral data," *Hum.-Comput. Interact.*, vol. 32, no. 5-6, pp. 447–510, Nov. 2017.

[10] H. Ji and R. Grishman, "Knowledge base population: Successful approaches and challenges," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 1148–1158.

[11] X. Zhang and H. Wang, "A joint model of intent determination and slot filling for spoken language understanding." in *IJCAI*, 2016, pp. 2993–2999.

[12] K. Yao, B. Peng, G. Zweig, D. Yu, X. Li, and F. Gao, "Recurrent conditional random field for language understanding," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 4077–4081.

[13] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge," in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 697–706.

[14] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, 2008, pp. 1247–1250.

[15] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang, "Knowledge vault: A web-scale approach to probabilistic knowledge fusion," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 601–610.

[16] R. Davis, H. Shrobe, and P. Szolovits, "What is a knowledge representation?" *AI magazine*, vol. 14, no. 1, p. 17, 1993.

[17] W. W. W. Consortium, "Rdf 1.1 concepts and abstract syntax," 2014.

[18] Q. Fu, J.-G. Lou, Y. Wang, and J. Li, "Execution anomaly detection in distributed systems through unstructured log analysis," in *Ninth IEEE International Conference on Data Mining*, ser. ICDM '09. IEEE, 2009, pp. 149–158.

[19] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[20] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *arXiv preprint arXiv:1607.04606*, 2016.

[21] Y. Li, Z. Yang, Y. Guo, and X. Chen, "Droidbot: A lightweight ui-guided test input generator for android," in *Proceedings of the 39th International Conference on Software Engineering Companion*, ser. ICSE-C '17. Piscataway, NJ, USA: IEEE Press, 2017, pp. 23–26.

[22] S. Vhaduri and C. Poellabauer, "Hierarchical cooperative discovery of personal places from location traces," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2017.

[23] R. Spolaor, E. D. Santo, and M. Conti, "Delta: Data extraction and logging tool for android," *IEEE Transactions on Mobile Computing*, vol. 17, no. 6, pp. 1289–1302, June 2018.

[24] Y. Li, F. Chen, T. J.-J. Li, Y. Guo, G. Huang, M. Fredrikson, Y. Agarwal, and J. I. Hong, "Privacystreams: Enabling transparency in personal data processing for mobile apps," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 1, no. 3, pp. 76:1–76:26, Sep. 2017.

[25] Y. Li, Y. Guo, and X. Chen, "Peruim: Understanding mobile application privacy with permission-ui mapping," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '16. New York, NY, USA: ACM, 2016, pp. 682–693.

[26] E. Fernandes, O. Riva, and S. Nath, "Appstract: on-the-fly app content semantics with better privacy," in *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*. ACM, 2016, pp. 361–374.

[27] A. Sahami Shirazi, N. Henze, T. Dingler, M. Pielot, D. Weber, and A. Schmidt, "Large-scale assessment of mobile notifications," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '14. New York, NY, USA: ACM, 2014, pp. 3055–3064.

[28] A. Mehrotra, V. Pejovic, J. Vermeulen, R. Hendley, and M. Musolesi, "My phone and me: Understanding people's receptivity to mobile notifications," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ser. CHI '16. New York, NY, USA: ACM, 2016, pp. 1021–1032.

[29] A. Mehrotra, R. Hendley, and M. Musolesi, "Prefminer: Mining user's preferences for intelligent mobile notification management," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '16. New York, NY, USA: ACM, 2016, pp. 1223–1234.

[30] B. Tabuenca, M. Kalz, S. Ternier, and M. Specht, "Stop and think: Exploring mobile notifications to foster reflective practice on meta-learning," *IEEE Transactions on Learning Technologies*, vol. 8, no. 1, pp. 124–135, Jan 2015.

[31] F. Bentley and K. Tollmar, "The power of mobile notifications to increase wellbeing logging behavior," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '13. New York, NY, USA: ACM, 2013, pp. 1095–1098.

[32] N. Kushmerick, D. S. Weld, and R. Doorenbos, "Wrapper induction for information extraction," 1997.

[33] I. Muslea, S. Minton, and C. A. Knoblock, "Hierarchical wrapper induction for semistructured information sources," *Autonomous Agents and Multi-Agent Systems*, vol. 4, no. 1-2, pp. 93–114, 2001.

[34] R. Baumgartner, S. Flesca, and G. Gottlob, "Visual web information extraction with lixto," in *Proceedings of 27th International Conference on Very Large Data Bases*, ser. VLDB '01. Morgan Kaufmann, 2001.

[35] V. Crescenzi, G. Mecca, P. Merialdo *et al.*, "Roadrunner: Towards automatic data extraction from large web sites," in *VLDB*, vol. 1, 2001, pp. 109–118.

[36] C.-H. Chang, C.-N. Hsu, and S.-C. Lui, "Automatic information extraction from semi-structured web pages by pattern discovery," *Decision Support Systems*, vol. 35, no. 1, pp. 129–147, 2003.