Multi-application Personalization: Data Propagation Evaluation on a Real-life Search Query Log

Abstract—In the field of multi-application personalization, several techniques have been proposed to support user modeling. None of them have sufficiently investigated the opportunity for a multi-application profile to evolve over time in order to avoid data inconsistency and the subsequent loss of income for website users and companies. In this paper, we propose a model addressing this issue and we focus in particular on user profile data propagation management. Data propagation is a way to reduce the amount of inconsistent user profile information over several applications, even in the case of temporary coalitions of applications as happens in Digital Ecosystems. To evaluate our model, we first extract user profiles using logs of the large real-life applications as happens in Digital Ecosystems. To evaluate our model, we first extract user profiles using logs of the large real-life applications as happens in Digital Ecosystems. In Section III we briefly describe current research in the field of personalization for future research and the conclusions.

I. INTRODUCTION

Nowadays, many different applications in different areas are concentrating on collecting information about users for service personalization. For this reason, applications organize user properties, preferences and assumptions based on the user state in user profiles. Each application manages user information independently from others, based on a specific user model. When user profile management is performed in an isolated way in each application, we are in presence of mono-application scenarios. This leads to data inconsistency between isolated user profiles.

In this paper, we propose a method that enables user profile information to evolve in a multi-application context based on user data propagation. We describe G-Profile, our multi-application user modeling system [1] and evaluate it in terms of data inconsistency reduction. To do this, we extract a global semantic representation of the AOL search query log [2] using a semantic clustering method [3] and simulate user data propagation along semantically related user information.

We refer to situations of collaborating applications – as happens in Digital Ecosystems – establishing data relationships between them for specific purposes or goals.

The rest of the paper is organized as follows: in Section II we briefly describe current research in the field of personalization in multi-application environments. In Section III we introduce G-Profile and the formalization of our model. We explain the concept of applications collaboration via G-Profile and the way user data are propagated. Section IV illustrates the user profile extraction process applied to the AOL search query log and the way user information is organized and regrouped based on semantics. In Section V we evaluate of our model in terms of data inconsistency reduction over the AOL search query log and in Section VI we present the main directions for future research and the conclusions.

II. BACKGROUND

Nowadays, distributed software environments are no longer static stand-alone applications [4]. In particular, Digital Ecosystems are self-organized collaborative environments based on multiple interacting applications. User modeling [4], [5] plays a crucial role in this scenario, and represents the basis for multi-application personalization [6].

From literature [7], [8], we outline two major approaches for user modeling in multi-application scenarios: (i) standardization-based user modeling, based on a top-down vision, defining some a priori – often centralized – standard whom all the involved applications have to comply [6], [9]–[16]; (ii) mediation-based user modeling, behaving in a bottom-up way, mapping different user model representations by the use of suitable mapping rules and/or meta-models [17].

In [18] authors suggest the integration of mediation-based techniques and standardization of user modeling based on Semantic Web technologies (WordNet [19], GUMO and UserML). The use of semantics for user models integration is proposed also in [7], [20].

All the techniques described before, address in different ways how to integrate a large number of available user model fragments for personalized service delivery, but fail in taking into account the possibility for user profiles to evolve over time, non considering the concept of user data propagation among them.

1The term ‘mediation’ has been introduced by Berkovsky et al. in [17]
III. G-PROFILE

The aim of G-Profile is to provide a general-purpose and flexible user modeling system for multi-application environments. With respect to techniques already proposed in literature, our approach is intended to address (i) the possibility for user profiles to evolve over time via data propagation in a (ii) secure and user-centered way.\footnote{Security and privacy issues are outside of the scope of this paper.}

G-Profile does not propose neither a specific reconciliation technique able to take into account all the possible user data representations in different applications, nor a standard user profile model. Instead, it is an abstract and flexible protocol that can be used to define some abstract mapping functions, based on the generic concept of mapping between user data among applications.\footnote{Semi-automatic generators of concrete mappings can be used.}

A. G-Profile Formalization

Being A the set of applications in a multi-application environment, each application $A_i$ ($i \in \{1, \ldots, n\}$ in $A$), manages a set $D^{A_i}$ of user attributes $a^{A_i}_t$ ($t \in \{1, \ldots, m\}$, $m \in \mathbb{N}$). For each user $u_x$ having a profile on the application $A_i$, each attribute $a^{A_i}_t$ has a value $v_t$ associated, forming the user profile element as a couple $e^{A_i}_{i,t} = (a^{A_i}_t, v_t)$, (i.e., attribute, value).

In our system each attribute can, from time to time, be involved as the source or the target attribute in a correlation with others. More specifically, since attributes are organized differently in each application $A_i$ depending on the adopted user model, they can be permuted in several source sets $S^{A_i}_t \subseteq S^{A_i}$, $1 \leq t \leq 2^{D^{A_i}}$. Each source set is composed of a number of $s^{A_i}_g$ source attributes, where $g \in \{1, \ldots, v\}$, $v \in \mathbb{N}$, $s^{A_i}_g \subseteq D^{A_i}$ and $S^{A_i} \subseteq D^{A_i}$.

In the same way, each attribute of the application $A_i$ can be a target attribute $t^{A_i}_h$ ($h \in \{1, \ldots, w\}$, $w \in \mathbb{N}$) belonging to the target set $T^{A_i}$, where $t^{A_i}_h \subseteq D^{A_i}$ and $T^{A_i} \subseteq D^{A_i}$.

We define a mapping $M^{A_i,A_j}$ between two applications $A_i$ and $A_j$, $i \neq j$, as the triple

$$M^{A_i,A_j} = (S^{A_i}, T^{A_i}, M^{A_i,A_j}) \quad (1)$$

where $M^{A_i,A_j} = \{m^{A_i,A_j}_k\}$ and $m^{A_i,A_j}_k$ is a mapping function between two applications associating to each source set $S^{A_i}_t$ a target attribute in $T^{A_j}$. Formally,

$$m^{A_i,A_j}_k : S^{A_i}_t \rightarrow t^{A_j}_h \quad (2)$$

We assume that the number of mapping functions is equal to the number of target attributes of $T^{A_j}$, i.e., $k = 1, \ldots, |T^{A_j}|$. Formally, $|M^{A_i,A_j}| = |T^{A_j}|$.

Let us indicate with $M$ the set of all the mappings $M^{A_i,A_j}$, $i \neq j$. It is possible to define a mapping graph $G$ as a combination $G = (M, E)$ of all the mappings in our environment: $G = G(M)$. More specifically, we define the mapping graph as a pair $G = (V, E)$ composed of (i) a set $V$ of nodes, (ii) a set $E$ of directed edges.

We define two different kinds of node, i.e., attribute nodes: n-att and function nodes: n-fun. Formally $V = V_{n-att} \cup V_{n-fun}$. In particular, we represent the elements of each $S^{A_i}_t$ and $T^{A_j}_h$ as n-att nodes, while the elements $m^{A_i,A_j}_k$ are represented as n-fun nodes. Formally, $S^{A_i}_t \in V_{n-att}$, $T^{A_j}_h \in V_{n-fun}$.

We also define a function $\text{nodeType} : V \rightarrow \{\text{n-att, n-fun}\}$ that retrieves the type of a certain node.

In the same way we define a function application $A_i \rightarrow \text{nodeType} : V_{n-att} \rightarrow A_i$ retrieving, for a certain attribute node, the application it belongs to. We represent an edge between two nodes $n_1$ and $n_2$ as $(n_1, n_2) \in E$, where

- $\forall (n_1, n_2) \in E \Rightarrow \text{nodeType}(n_1) \neq \text{nodeType}(n_2)$,
- if $(n_1, n_2) \in E$, $(n_2, n_3) \in E$ and $\text{nodeType}(n_2) = \text{nodeType}(n_3)$ then $(n_1, n_3) \in E$.

In order to build the mapping graph, we define Algorithm 1.

B. Data Propagation

Let us consider two applications $A_i$ and $A_j$, $i \neq j$, connected via a mapping $M^{A_i,A_j}$. Let us suppose that a modification occurs on a source object $s^{A_i}_g$ in $S^{A_i}$, where $S^{A_i}$ is connected via a mapping function $m^{A_i,A_j}_k$, to an element $t^{A_j}_h$. It is our idea that the modification on $s^{A_i}_g$ is propagated – via G-Profile – to $t^{A_j}_h$ only if certain conditions imposed by the application $A_j$ hold. To do this, every time a modification takes place on $s^{A_i}_g$, a set $(A)_{s^{A_i}_g} = \{(\alpha)_{s^{A_i}_g}^0, (\alpha)_{s^{A_i}_g}^1, (\alpha)_{s^{A_i}_g}^a, (\alpha)_{s^{A_i}_g}^1, \ldots, (\alpha)_{s^{A_i}_g}^{A_j}\}$ of propagation attributes, connected to $s^{A_i}_g$, is transmitted to G-Profile. This set contains always the identification of the application $A_i$ at the origin of the modification. This information is retained...
Algorithm 1 Construction of the mapping graph $G$.

Input: All the mappings $M_{A_i,A_j} \in M$

Output: $G$

1: $G = \emptyset$
2: for all $M_{A_i,A_j} \in M$ do
3: for all $m_{k,A_i,A_j} \in M_{A_i,A_j}$ do
4: add $m_{k,A_i,A_j}$ to $V$
5: for all $s_y^{A_i} \in S^{A_i}$ do
6: if $s_y^{A_i} \notin V$ then
7: add $s_y^{A_i}$ to $V$
8: end if
9: add $(s_y^{A_i}, m_{k,A_i,A_j})$ to $E$
10: end for
11: if $t_h^{A_j} \notin V$ then
12: add $t_h^{A_j}$ to $V$
13: end if
14: add $(m_{k,A_i,A_j}, t_h^{A_j})$ to $E$
15: end for
16: end for

by the attribute denoted as $(\alpha)^o$, that we call origin of the modification. In the same way, the set always contains the absolute modification time attribute, denoted as $(\alpha)^t$. It represents the instant (in absolute terms) wherein the original modification occurs.

Each application $A_j$ defines, for each of its target elements $t_h^{A_j}$, a set $(K)^j_{h} = \{(\kappa_1)^j_{h}, (\kappa_2)^j_{h}, \ldots, (\kappa_n)^j_{h}\}$ of propagation conditions. Each propagation condition $(\kappa_i)^j_{h}$ is a boolean predicate which can be based on the set $(A_i)^{A_j}$ or directly on $A_j$’s rules.

This way, we define a boolean function that we call mapping activation function $f((K)^j_{h}) \rightarrow I$, where $I = \{0, 1\}$, enabling the propagation of a change on $s_y^{A_i}$ to $t_h^{A_j}$ using $m_{k,A_i,A_j}$ if $f((K)^j_{h}) = 1$.

The concepts of propagation attributes, propagation conditions and mapping activation function, constitute conditional data propagation. Via the conditional propagation of data, we are able to control recursive, partial and contextual data propagation among applications. Contextual data propagation is particularly useful in Digital Ecosystems, where G-Profile can help in managing propagation only between specific applications, part of a specific ecosystem. The treatment of problems connected to recursive data propagation, notably (i) cycles and (ii) parallelism are out of the scope of this paper, but details are given in [1].

IV. BOTTOM-UP DATA EXTRACTION AND MAPPING

Finding real user profiles associated to a concrete multi-application environment definitely proves difficult, due in particular to applications reluctance in releasing user profiles, because of privacy concerns and the high value of data. For this reason, in order to evaluate G-Profile on a realistic scenario, we extract user profiles from a real-life search query log in a bottom-up way, taking semantics into account.

More specifically, we act on the AOL search query log. This log is constituted by a set of user search queries followed by the domain of the URL selected by the user among the results. We interpret queries as topics of interest of the users, connected to specific domains. Based on this interpretation, we consider a user profile as being constituted by all the topics of interest connected to a specific domain for a particular user. Each domain represents an application in the G-Profile sense; in the same manner, each topic of interest represents an attribute connected to a specific application.

On the base of these assumptions, by evaluating the semantic distance between topics of interest belonging to different domains, we obtain a measure of semantic relatedness between attributes belonging to different applications. By thresholding this measure, we can propose possible mappings between attributes in the G-Profile sense, and obtain a bottom-up mapping graph on which we are able to evaluate user data propagation in G-Profile.

Figure 2 illustrate our three-step technique to produce the mapping graph described before. Having as input the AOL search query log, the first step produces a semantically enhanced global reference for the whole log. It is a data structure organizing, for each domain, the query terms in a taxonomy, built over the terms used in keyword search by means of the WordNet lexical database [19]. The second step considers this global reference as a platform on which the user interests are (i) identified and (ii) clustered. Finally, the third step

![Fig. 2. Schema of the bottom-up mapping graph generation technique.](image-url)
consists in (i) electing a representative concept for each cluster (it corresponds to an attribute in the G-Profile sense), (ii) evaluating semantic distance between representative concepts (attributes) belonging to different domains (applications) and (iii) mapping representative concepts (attributes) based on a level of semantic similarity between them.

A. Global Semantic Representation

Aim of this step, is to supply the search query log with semantics, i.e., to take into account the meanings of log terms and the semantic relations between them. WordNet provides us with information about terms and the hypernymy relation (is-a), that we use as a basic structural characteristic that defines a hierarchical order between terms (Figure 3).

In order to evaluate the semantic distance between two terms, we first introduce a weight associated to the individual “is-a” links. This weight is defined as a decreasing function of the semantic proximity between the parent and the child, i.e., the higher the weight, the less related the terms. As all relations contained in the taxonomy are of type “is-a”, the nodes go from the most general at the top to the most specific at the bottom. Therefore, two connected terms at the bottom of the taxonomy are more closely related than two connected terms at the top. The weighting function should thus be decreasing with respect to the level of the terms. Based on this, we define the weight function

$$W(x, y) = \frac{1}{l(y)}$$

where $x$ and $y$ are two terms related by the direct relation $y$ “is-a” $x$ and $l$ is the function that returns the level of $y$.

In order to be able to compare every couple of terms of the taxonomy, we introduce a distance function

$$Distance(x, y) = \begin{cases} \sum_{i=l(x)+1}^{l(y)} \frac{1}{i} & \text{if } x \rightarrow y \text{ is a straight path} \\ \sum_{i=l(c)+1}^{l(y)} \frac{1}{i} + \sum_{i=l(c)+1}^{l(x)} \frac{1}{i} & \text{otherwise} \end{cases}$$

($l$ being the level function and $c$ the common hypernym of terms $x$ and $y$) that depends on the weights of the edges composing the path between the terms (Figure 4)

B. User Interests Extraction

The problem of user interests extraction, can be technically defined as a query terms clustering problem, usually treated [21], [22] without considering the semantic relations between query terms. On the contrary, we propose an algorithm based on the semantic distance defined above, defining a cluster as a set of query terms such that the distance between each pair of query terms of the set is inferior to a predefined threshold.

Algorithm 2 is designed to take advantage of the decreasing property of our distance function, while none of the reviewed algorithms is adapted to this property. First, it starts by finding the deepest term $e_d$ in the taxonomy (the goal is to get the most specialized terms in one cluster). Second, it tries to find the closest terms to this term that respect the threshold by switching in two dimensions, height (parents) and width (children) of the taxonomy. Thus, first it checks the distance between $e_d$ and the first parent term with respect to the threshold using the function $\text{cluster}_\text{up}$. Then, if the condition is satisfied, it uses $\text{cluster}_\text{down}$ to similarly check the children terms related to that parent. The process is recursively applied on the next parents until there is no term that respects the threshold with the initial term. The result of this iteration is one cluster; all terms of the cluster are excluded from the next iterations. In the next iteration, the deepest term is sought again. The algorithm iterates on the rest of the terms until they are all clustered.

C. User Interests Mapping

This step relates the previously identified clusters by means of possible mappings. We first elect the representative terms of the clusters which will be used for comparison. The election process is done according to the level of abstraction of terms. We consider a representative term to be the most abstract term in the cluster. The comparison is done by using the distance (4). Thus, we define a mapping function between the cluster nodes $C_i^{D_1}$ and $C_j^{D_2}$ according to G-profile, if :

$$M(C_i^{D_1}, C_j^{D_2}) = 1 \text{ if } Distance(\text{RC}_k^{D_1}, \text{RC}_l^{D_2}) < \alpha$$

(5)

where $\text{RC}_j$ and $\text{RC}_j$ are their correspondent representatives and $\alpha$ is a predefined threshold to realize a mapping between two cluster. Once obtained mappings among clusters this way, we generate the mapping graph according to Algorithm 1.

V. Evaluation

We evaluate the advantages of using G-Profile in terms of data inconsistency reduction, in the multi-application environment represented by the global taxonomy extracted from the
Algorithm 2 Query terms clustering algorithm.

Input: $T$ (taxonomy with weighted links), $E = \{e_0, e_1, \ldots\}$ (set of query terms (nodes)), $D$ (distance function), $\alpha$ (threshold value)

Output: $E$

1. $C = \{\emptyset\}$ (set of clusters)
2. $c_i = \emptyset$ ($c_i \in C$)
3. while not empty($E$) do
4.   $e_d = \text{deepest}(E)$ (find the deepest term)
5.   $c_i = c_i \cup \{e_d\}$ (init. $c_i$ with the deepest term)
6.   CLUSTER UP($e_d, \text{PARENT}(e_d)$)
7.   $C = C \cup \{c_i\}$
8.   $E = E - \{c_i\}$
9. end while

10. function CLUSTER UP(predecessor, $e$)
11.   if $D(e_d, e) \leq \alpha$ then
12.     while HAS CHILDREN($e$) do
13.       if CHILD OF ($e$) $\neq$ predecessor then
14.         CLUSTER DOWN(PULL CHILD OF($e$))
15.       end if
16.     end while
17.     $c_i = c_i \cup \{e\}$
18.   end if
19. end function

20. function CLUSTER DOWN($e$)
21.   if $D(e_d, e) \leq \alpha$ then
22.     while HAS CHILDREN($e$) do
23.       CLUSTER DOWN(PULL CHILD OF($e$))
24.     end while
25.     $c_i = c_i \cup \{e\}$
26.   end if
27. end function

AOL search query log. Each domain in the taxonomy represents an application, while each topic of interest associated to a domain, represents an attribute detained by an application.

We simulate that, each modification occurring on an attribute, possible inconsistencies emerge (over the time) in user data belonging to other applications referring to the modified user data. As we have explained at the beginning of the paper, each user manages different user profiles in different applications and she usually manually modifies the same data in different applications. Before the user has updated all her data, they remain inconsistent for a certain period of time. We demonstrate that, given the same period of time, the use of G-Profile leads to a smaller number of inconsistencies in the environment, thanks to data propagation.

### A. Mapping Graph Generation

We choose, for the bottom-up generation of the graph, a number of domains (applications) equal to 100, selecting the domains referred by the higher number of users and for which the users have done the higher number of queries. This way, we obtain a number of attributes in the vicinity of 200K. In order to generate mappings between attributes, we choose two thresholds to apply to (5). We choose $\alpha = 0.85$ for unidirectional mappings and $\alpha = 0.95$ for bidirectional mappings. The choice of a threshold in the vicinity of 1 for bidirectional mappings is due to the fact that we expect that attributes involved in a bidirectional correlation represent the same concept. Having chosen these thresholds, permits to generate a mapping graph with about 400K mappings.

### B. Data Propagation Evaluation

For evaluation purposes, we simulate a G-Profile and a non-G-Profile behavior of the system. We suppose that, during each given period of time, some modifications occur on certain attributes in the system. Due to these modifications, attributes that are directly or indirectly mapped, according to G-Profile, with modified ones, become inconsistent. In the case of the G-Profile aware simulation, at each cycle we propagate the modifications to the directly mapped attributes making them consistent. If no mapping among attributes is provided (the non-G-Profile aware simulation), we set random updates of the inconsistent nodes, representing the manual change of these attributes over the time. The parameters we can act on are:

- **interModCycleCount**: the duration of the modification time period
- **timeStampCount**: the duration of the simulation.

The number of modifications can be updated at each modification time period following a Poisson distribution: $f(k, \lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$. We can act on the `poissonMean` parameter representing the $\lambda$ property of the distribution. This way, we represent the fact that the number of modifications in a system varies in intensity during the time.

Even the duration of the modification time period can be updated after each modification time period following a Poisson distribution, via the `cyclePoissonMean` parameter. In this case, we represent the fact that the frequency of modifications varies in intensity during the time.

Within the chosen number of modifications in a given modification time period, it is possible to choose the attributes to modify in a non-random way, accordingly to the Zipf’s law. In probability theory and statistics, it refers to a class of discrete probability distributions. This law is used to describe phenomena where large events are rare, but small ones quite common. In our evaluation, we use Zipf’s law to describe the fact that a large number of attributes nodes is modified only occasionally whilst few attribute nodes are modified frequently. We can act on the `zipfSkew` parameter, corresponding to the $s$ value.

We set parameters as follows: `timeStampCount`: 150, `poissonMean` = 10.0 and `zipfSkew` = 0.5. Figure 5 shows the positive effects of the use of G-Profile in lowering the number of potential inconsistencies in a multi-application environment due to modifications in user profiles over a period of time. In particular, in (a) we show the effects of G-Profile with `interModCycleCount` = 5. In (b) we set a higher duration

\[ f(r; s, n) = \frac{1}{\sum_{r=1}^{n} \frac{1}{r^s}} \]
for the modification time period: $\text{interModCycleCount} = 10$. In (c) we vary the duration of each modification time period setting $\text{cyclePoissonMean} = 10.0$.

![Fig. 5. Data propagation effects with and without G-Profile.](image)

In all the configurations (Figure 5 (d)), emerges that using G-Profile leads to a quasi-complete inconsistency zeroing depending on the $\text{interModCycleCount}$ parameter value chosen. The bigger the value of this parameter, the lower is the number of inconsistencies and vice versa. In particular, independently from the chosen duration of the modification time period, G-Profile shows, applied to the specific simulation, a number of inconsistencies between $0 < i < 2000$, while in non-G-Profile environments, the number of inconsistencies is continuously growing.

VI. CONCLUSIONS AND FURTHER RESEARCH

In recent years, several approaches have been proposed in different fields to solve the problem of multi-application personalization. The focus is gradually shifted from the model itself to the process of modeling. G-Profile aims to be a flexible multi-application user modeling system, able to address interpretability problems of collaborative distributed environments (such as Digital Ecosystems) and in particular to guarantee the evolution of user profiles and security and privacy issues in these environments, aspects that have not been sufficiently considered up to now. In this paper we presented our technique, with particular reference to the process of user profile data propagation. We showed, on a concrete scenario, the efficiency of G-Profile in lowering the number of inconsistencies of user data in multi-application environments. We are now working on aspects connected to partial and contextual data propagation, and on how to make more realistic their evaluation.

REFERENCES


tf_202325v010101p.pdf.


