# Mirror, Mirror, on the Web, Which Is the Most Reputable Service of Them All?

## A Domain-Aware and Reputation-Aware Method for Service Recommendation

Keman Huang[1], Jinhui Yao[2], Yushun Fan[1], Wei Tan[3],
Surya Nepal[4], Yayu Ni[1], and Shiping Chen[4]

[1] Department of Automation, Tsinghua University, Beijing 100084, China
[2] School of Electrical and Information Engineering, University of Sydney, Australia
[3] IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598, USA
[4] Information Engineering Laboratory, CSIRO ICT Centre, Australia
hkm09@mails.tsinghua.edu.cn, jinhui.yao@gmail.com,
fanyus@tsinghua.edu.cn, wtan@us.ibm.com, surya.nepal@csiro.au,
nyy07@mails.tsinghua.edu.cn, shiping.chen@csiro.au

**Abstract.** With the wide adoption of service and cloud computing, nowadays we observe a rapidly increasing number of services and their compositions, resulting in a complex and evolving service ecosystem. Facing a huge number of services with similar functionalities, how to identify the core services in different domains and recommend the trustworthy ones for developers is an important issue for the promotion of the service ecosystem. In this paper, we present a heterogeneous network model, and then a unified reputation propagation (URP) framework is introduced to calculate the global reputation of entities in the ecosystem. Furthermore, the topic model based on Latent Dirichlet Allocation (LDA) is used to cluster the services into specific domains. Combining URP with the topic model, we re-rank services' reputations to distinguish the core services so as to recommend trustworthy domain-aware services. Experiments on ProgrammableWeb data show that, by fusing the heterogeneous network model and the topic model, we gain a 66.67% improvement on top20 precision and 20%~ 30% improvement on long tail (top200~top500) precision. Furthermore, the reputation and domain-aware recommendation method gains a 118.54% improvement on top10 precision.

**Keywords:** Heterogeneous Network, Reputation Propagation, Topic Model, Service Recommendation, Service Ecosystem.

## 1 Introduction

With the wide adoption of service and cloud computing, nowadays we observe a rapidly increasing number of services and their compositions (mashups, workflows) [1]. Internet companies such as Google, Flickr, and Facebook publicly provide the APIs of their services, which effectively motivates individual developers to combine available services (e.g. web services, web APIs) into innovative service-compositions/

mashups as a value-add to these existing web services. As a consequence, several domain-specific or general purpose online service ecosystem, such as ProgrammableWeb[1], myExpriment[2] and Biocatalogue[3], have emerged and collected a rapidly increasing number of services and their compositions in recent years. As atomic services are composed into composite ones, they are not isolated but influenced with each other. Thus the services, compositions (mashups, workflows), the service providers and the composition developers together form a complex and evolving service ecosystem. When constructing a composition in the service ecosystem, the straightforward method for the developers is to refer to the related domain and then select the trustworthy services to compose. However, it is difficult to guarantee this trustworthiness in the real practice.

First of all, the querying power in the recent service ecosystems is usually preliminary [2]. Taking ProgrammableWeb as an example, services are registered into a specific category with only a simple word such as "Bookmarks", "Search" and "Social", etc. However, some services naturally belong to multiple domains as they offer different domain-specific functionalities. For example, "del.icio.us" is a famous service for the social bookmarks which also can be used to store the users' bookmarks online and search the bookmarks by tags. Thus it not only belongs to the category "Bookmarks" and "Social" , but also the category "Database" and "Search". Thus in this paper, a topic model based on Latent Dirichlet Allocation (LDA) [3] is used to cluster the services so that we can assign the services into different domains.

Secondly, facing the huge number of services, the developers need to select the desirable services against many other alternatives which are similar to one another. Therefore, the ecosystem should not only list services in different domains but also provide guidelines for selecting the trustworthy ones based on their performance in the past. We define *trust* as the belief that a user has regarded the intention and capability of a service/mashup to behave as expected. We use reputation as a mechanism of establishing the belief about a service's ability to deliver a certain service level objective [4, 14]. The notion underpinning the reputation-based trust models is to capture consumers' perception of the consumed service and use it to evaluate the reputation of the service [5, 20]. Many researches use the quality of service (Qos) combining with the Collaborative Filtering (CF) to calculate the reputation of the services [6-9]. However, it is resource-intensive and sometime impossible to fetch the Qos of the services over time, especially when considering the different Qos metrics that can be applied for different types of services that are deployed remotely. Taking ProgrammableWeb as an example, there is no information about the Qos for most of services. Fortunately, the historical usage information embedded with the related consumers' experiences [10] and the collective perception from the developers can be used to calculate the reputation of services in the ecosystem. In this paper, we present a unified reputation propagation (URP) method to calculate services' reputation so as to facilitate trust-aware recommendations. Furthermore, different services with unique

[1] `http://www.programmableweb.com`
[2] `http://www.myexperiment.org/`
[3] `http://www.biocatalogue.org/`

functionalities have specific areas where they can perform better than others [11]. Therefore, combining the services' reputation and the LDA model, we re-rank the services to get a domain-aware recommendation for their specific functionality domains. Based on these, the main contribution of this paper is as follow:

1) We propose a heterogeneous network model of the service ecosystem and the unified reputation propagation (URP) framework is used to calculate the global reputation of various entities in the ecosystem.

2) The LDA model is used to analyze the services in the ecosystem and re-cluster them into different domains. Combing the LDA and the URP-based reputation-ranking method, for the first time, we offer the reputation- and domain-aware service recommendation.

3) Experiments on the real-world dataset, i.e., ProgrammableWeb, show that our method gain a 66.67% improvement on top20 precision and 20%~ 30% improvement on long tail (top200~top500) precision, compared to the methods based on the homogeneous network. Furthermore, the reputation and topic aware recommendation method gains a 118.54% improvement on top10 precision, compared to the domain-only method. Thus our approach can effectively offer trustworthy recommendation for developers.

The rest of the paper is organized as follows. Section 2 introduces a heterogeneous network model and the reputation propagation framework to calculate the global reputation. Section 3 shows the domain-aware recommend method which combines the topic model and the global reputation. Section 4 reports our empirical experiments on the real-world data ProgrammableWeb. Section 5 discusses the related work and Section 6 concludes this paper.

## 2 Unified Reputation Network Model

### 2.1 Heterogeneous Network Model

In the service ecosystem, service providers publish services into the ecosystem and then those services are classified into different domains based on their functionalities. Composition developers will choose one or more services and combine them into a composition (i.e. mashup) and publish it into the ecosystem which will be used by consumers. Throughout this paper, we will use "composition" and "mashup" interchangeably as they both combine atomic services to provide added value. Considering these, we can model the ecosystem into a heterogeneous network model which contains developers, mashups, services and providers as well as the relationships among them. Figure 1 shows the schematic diagram of the heterogeneous network and we can formalize it as follow:

**Definition 1 (Heterogeneous Network for a Service Ecosystem).** A service ecosystem is a heterogeneous network $G=(V,\mathrm{E})$ where $V=\{De,Ma,Se,Pr\}$ refers to the four different types of entities in the ecosystem. $De$ refers to all the developers who publish at least one mashup in the ecosystem. $Ma$ refers to all the mashups. $Se$ refers to the services and $Pr$ refers to the service providers. $\mathrm{E}=\{D,Y,P\}$ refers to the three kinds of relationships among the entities. $D$ refers to the developer-mashup network, $Y$ is the mashup-service network and $P$ is the provider-service network. These three networks can be defined in matrix as follows:

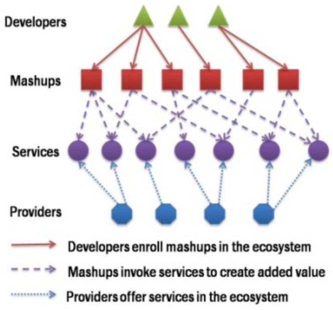
Developers
Mashups
Services
Providers
Developers enroll mashups in the ecosystem
Mashups invoke services to create added value
Providers offer services in the ecosystem

**Fig. 1.** The heterogeneous network model for service ecosystem which including developers, mashups, services, providers as well as the three kinds of relationships among them

**Definition 2 (Developer-mashup Network)** The developer-mashup network is used to present the publish relationship between developers $De$ and mashups $Ma$. It is denoted by a $n \times m$ matrix $D = \left[d_{ij}\right]_{n \times m}$ and the element is $d_{ij} = \begin{cases} 1 \text{ if } De_i \text{ develops } Ma_j \\ 0 \quad \text{otherwise} \end{cases}$ where $n$ refers to the number of developers and $m$ is the number of mashups.

**Definition 3 (Mashup-service Network)** The mashup-service network is used to present the invoking relationship between mashups $Ma$ and services $Se$. It is denoted by a $m \times s$ matrix $Y = \left[y_{jk}\right]_{m \times s}$ and the element is $y_{jk} = \begin{cases} 1 \text{ if } Ma_j \text{ invokes } Se_k \\ 0 \quad \text{otherwise} \end{cases}$ where $m$ is the number of mashups and $s$ is the number of services.

**Definition 4 (Provider-service Network)** The provider-service network is used to present the supply relationship between services and providers. It is denoted by a $p \times s$ matrix $P = \left[p_{ok}\right]_{p \times s}$ and the element $p_{ok} = \begin{cases} 1 \text{ if } Pr_o \text{ provides } Se_k \\ 0 \quad \text{otherwise} \end{cases}$ and $p$ is the number of providers.

Furthermore, based on the definitions shown above, we can get the derivations:

**Definition 5 (Service Co-occurrence Network)** The service co-occurrence network is denoted by a $s \times s$ matrix $S = [f_{kl}]_{s \times s}$ in which $f_{kk}$ = the number of mashups the service $k$ is invoked and $f_{kl}$ is the number that service $k$ and service $l$ are used together in the same mashup. We denote the main diagonal as $\Lambda = [f_{kk}]_{s \times s}$ and it is easy to get that:

$$f_{kk} = \sum_j y_{jk} \tag{1}$$

**Definition 6:** Developer reputation is a $n \times 1$ vector $R_d = [rd_i]_{n\times 1}$ and the element $rd_i$ refers to the reputation of developer $i$.

**Definition 7:** Mashup reputation is a $m \times 1$ vector $R_y = \left[ry_j\right]_{m\times 1}$ and the element $ry_i$ refers to the reputation of mashup $j$.

**Definition 8:** Service reputation is a $s \times 1$ vector $R_x = [rx_k]_{s\times 1}$ and the element $rx_k$ refers to the reputation of service $k$.

**Definition 9:** Provider reputation is a $p \times 1$ vector $R_p = [rp_o]_{p\times 1}$ and the element $rp_o$ refers to the reputation of provider $o$.

## 2.2 Unified Reputation Propagation Model

It is troublesome to calculate the reputation of the four entities at once in the ecosystem as they are affecting each other simultaneously. However, there are a few basic assumptions that we can take to simplify the calculation:

*Assumption 1.* **Highly reputable providers will offer many highly reputable services which are invoked in many highly reputable mashups.**

*Assumption 2.* **Highly reputable developers will develop many highly reputable mashups which invoke the highly reputable services.**

As we known, the evolution of the ecosystem is driven by the enrolling of new mashups and services. However, the services which are never invoked will not affect other services' reputations, including the newly registered one. So the propagation of the reputation in the ecosystem will be activated by the mashup. Thus we get the third assumption in this paper:

*Assumption 3.* **The reputation of mashups activates the reputation propagation process in the ecosystem.**

Based on these assumptions, we have a simple illustration of the reputation propagation model as Figure 2.

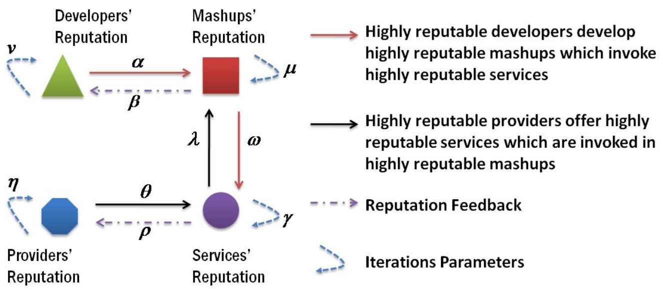
Developers' Reputation
Mashups' Reputation
Providers' Reputation
Services' Reputation
ν
α
β
μ
λ
ω
η
θ
ρ
γ
Highly reputable developers develop highly reputable mashups which invoke highly reputable services
Highly reputable providers offer highly reputable services which are invoked in highly reputable mashups
Reputation Feedback
Iterations Parameters

**Fig. 2.** The unified reputation propagation model for the service ecosystem

**Mashup Reputation**

The reputation of the mashup is decided by the reputation of the services it invoked and the reputation of its developer. Thus we can get the iterative equation as:

$$R_y^+ \leftarrow \mu R_y + \alpha D^T R_d + \lambda Y \Lambda^{-1} R_x + \xi_y \tag{2}$$

Here $\mu + \alpha + \lambda \leq 1, \mu \geq 0, \alpha \geq 0, \lambda \geq 0$ . $\alpha D^T R_d$ refers to the reputation from its developers; $\lambda Y \Lambda^{-1} R_x$ refers to the reputation from its invoking services, $\xi_y$ refers to the random factors and $\mu R_y$ is used for iteration. Then the reputation is normalized:

$$R_y^+ \leftarrow \frac{R_y^+}{\mathbf{1} \bullet R_y^+} \tag{3}$$

**Developer Reputation**

The reputation of the developer comes from the mashups he/she has ever published in the past. Intuitively, the higher the reputation his/her mashups have, the higher the reputation he/she will gain. Thus the iterative equation can be defined as:

$$R_d^+ \leftarrow v R_d + \beta D R_y^+ + \xi_d \tag{4}$$

Here $v + \beta \leq 1, v \geq 0, \beta \geq 0$ . $\beta D R_y^+$ refers to reputation from the mashups the developers have ever published and $\xi_d$ refers to the random factors. $v R_d$ refers to iteration from the last step. Then the reputation is normalized:

$$R_d^+ \leftarrow \frac{R_d^+}{\mathbf{1} \bullet R_d^+} \tag{5}$$

**Service Reputation**

The service reputation comes from the reputation of the mashups it has been invoked in and the reputation of its providers. We define the iterative equation as:

$$R_x^+ \leftarrow (\gamma - \lambda \omega) R_x + \omega Y^T R_y^+ + \theta P^T R_p^+ + \xi_x \tag{6}$$

Here $\gamma + \omega + \theta \leq 1, \gamma, \omega, \theta \geq 0$ . $\omega Y^T R_y^+$ refers to the reputation updating from mashups, $\theta P^T R_p^+$ refers to the reputation from the providers and $\xi_x$ refers to the random factors. $(\gamma - \lambda \omega) R_x$ is used for the iteration. Then we normalize the vector as:

$$R_x^+ \leftarrow \frac{R_x^+}{\mathbf{1} \bullet R_x^+} \tag{7}$$

**Provider Reputation**

The reputation of the providers comes from the services he/she published in the ecosystem. Thus the iterative equation is as follow:

$$R_p^+ \leftarrow \eta R_p + \rho P R_x + \xi_p \tag{8}$$

Here $\eta + \rho \le 1, \eta, \rho \ge 0$. $\rho PR_x$ refers to the reputation from services, $\xi_p$ refers to the random factors and $\eta R_p$ is used for iteration. Then we normalize the vector:

$$R_p^+ \leftarrow \frac{R_p^+}{\mathbf{1} \bullet R_p^+} \tag{9}$$

## 2.3 Model Simplification

Until now, we have proposed the iteration method for the reputation propagation so that we can gain the global reputation of the entities in the ecosystem. By setting different parameter combinations, we can derive three different propagation methods:

**Top-Popularity Reputation (TR) Model**
Here we set $\alpha = 0, \mu = 1, \lambda = 0, \omega = 1, \theta = 0, \gamma = 0$ and set the random factor $\xi_y = 0, \xi_x = 0$ We can easily get that:

$$R_y = R_y^0 \quad R_x = \frac{Y^T R_y^0}{\mathbf{1} \bullet Y^T R_y^0} = \frac{1}{C} Y^T R_y^0 \tag{10}$$

Where $R_y^0$ refers to the initial reputation of mashup and $C = \mathbf{1} \bullet Y^T R_y^0$ will be a constant for different services in the ecosystem. Furthermore, we set $R_y^0 = \left[ {}^1\!/_m \right]_{m \times 1}$ which means that the initial reputation for each mashup is equivalent, then we get:

$$R_x(k) \propto \sum\nolimits_j y_{jk} = f_{kk} \tag{11}$$

This means that the reputations of services are just based on its used frequency.

**Page-Rank-Based Reputation (PR) Model**
Here we set $\alpha = 0, \mu = 0, \lambda = 1, \theta = 0, \gamma = 0$ and the random factor for mashup's reputation is set as 0, we can easily get that:

$$R_y^+ = \frac{Y \Lambda^{-1} R_x}{\mathbf{1} \bullet Y \Lambda^{-1} R_x} \tag{12}$$

As $\mathbf{1} \bullet Y \Lambda^{-1} R_x = \sum\nolimits_k rx_k = 1$ we can further get that:

$$\begin{aligned} R_x^+ &= -\omega R_x + \omega Y^T R_y^+ + \xi_x = -\omega R_x + \frac{\omega Y^T Y \Lambda^{-1} R_x}{\mathbf{1} \bullet Y \Lambda^{-1} R_x} + \xi_x \\ &= \omega (Y^T Y \Lambda^{-1} - I) R_x + \xi_x \end{aligned} \tag{13}$$

Setting $\xi_x = (1-\omega) \left[ \frac{1}{n} \right]_{n \times 1}$ we can get that:

$$R_x^+ = \omega(Y^T Y \Lambda^{-1} - I)R_x + (1-\omega)\left[\frac{1}{n}\right]_{n\times 1} \tag{14}$$

$$rx_i^+ = \omega \sum\nolimits_k \frac{f_{ki}}{f_{kk}} rx_k + \frac{(1-\omega)}{n}, i \neq k \tag{15}$$

Apparently, the reputation of the service is based on the reputation of its neighbors. In this case, our model can be reduced into a page-rank [12] algorithm method just based on the co-occurrence service network.

**Developer-Related Reputation (DR) Model**
Here we set $\beta = 1, \nu = 0, \mu = 1-\alpha, \lambda = 0, \omega = 1, \gamma = 0, \theta = 0$ and $\xi_y, \xi_d, \xi_x = 0$ which means the reputation of the developers just come from the reputation of mashups, the reputation of the services is decided by the mashups it has been invoked in.

$$R_y^+ = \frac{(1-\alpha)R_y + \alpha D^T R_d}{\mathbf{1}\bullet((1-\alpha)R_y + \alpha D^T R_d)} = (1-\alpha)R_y + \alpha D^T R_d \tag{16}$$

$$R_d^+ = \frac{DR_y^+}{\mathbf{1}\bullet DR_y^+} = DR_y^+ \quad R_x^+ = \frac{Y^T R_y^+}{\mathbf{1}\bullet Y^T R_y^+} \tag{17}$$

Furthermore we can get the reputation for mashups and developers as:

$$R_y^t = (I - \alpha(I - D^T D))^t R_y^0 \quad R_d^t = D(I - \alpha(I - D^T D))^t R_y^0 \tag{18}$$

Here $t$ refers to the number of iterations. If we set $\alpha = 0$, this model will reduce to the TR model.

## 2.4 Initial Strategy

From the discussion above, we can observe that the global reputation is related to the initial reputation of each mashup. We define two initial reputations as:

**1) Equivalent Initial Mashup Reputation (EI):** The hypothesis here is that the initial reputation for each mashup is equivalent. We can set that $ER_y^0 = \left[ {}^{1}\!/_{m} \right]_{m\times 1}$

**2) Popularity-based Initial Mashup Reputation (PI):** The hypothesis here is that highly reputable mashups will attract consumers' high attention, which will be reflected in their rating and visited number. We define the popularity of the mashup as the product of its rating and visited number. Then we use the normalized popularity as the initial reputation. Thus we can get that $PR_y^0 = \left[ prx_i^0 \right]_{m\times 1}$

$$prx_i^0 = \frac{Rate(Ma_i) \times Visited(Ma_i)}{\sum_k Rate(Ma_k) \times Visited(Ma_k)} \tag{19}$$

## 3 Domain-Aware and Reputation-Aware Recommendation

As we discussed in the introduction, the current categorization method for the service ecosystem such as ProgrammableWeb is rather preliminary. Topic model based on LDA performs well for the document analysis, thus we will firstly employ the LDA to analyze the context (tags, description, summary, etc) of the services in the ecosystem and extract different topics from the context. Each topic is considered as a domain and each service will be affixed with the affiliation degree to the domain. Then for each domain, the services with a top-k affiliate degree will be selected and considered as the related services in the domain. Based on the URP framework, we can get the global reputation for each service in the ecosystem which reflects the collective perception from the historical information. Thus we can re-rank the services in each domain by the global reputation so that we can get the top trustworthy services for the developers in each domain. Table 1 shows the detail of our method for the domain-aware and reputation-aware recommendation.

From the algorithm we can see that the topic model is used to cluster the services by their context so that we can get the most related services; then the global reputation is used to re-rank the services by their reputation so that the trustworthy services can be recommended. Thus this method can recommend the highly trustworthy services in each domain for the developers.

**Table 1.** Combining Topic Model and Reputation for Recommendation

```
Algorithm: Topic Model with Reputation for Recommend
Input:
(1) Service list $Se$ with the global reputation for each service $R_x$
(2) Topic/Domain Number: $T$
(3) Top number of services for each topic: $k$ ; Parameter $q$
Output:
(1) Top-k services for each Topic
01. Running LDA method to extract the $T$ topics in the service list
02. For each topic, sort the services by its affiliate degree then get the top $qk$ services
03. Sort the $qk$ services by the reputation $R_x$ and then get the top-k services for each topic
```

## 4 Empirical Study on ProgrammableWeb

### 4.1 Experiment Data Set

To the best of our knowledge, ProgrammableWeb is by far the largest online repository of web services, and their mashups. In this paper, we obtain the data regarding services and compositions from June 2005 to March 2013. Each service contains the information such as name, provider, category, publication date, summary and description; each mashup contains the information such as name, creation data, developer, the list of services in it, its description and its visited number as well as the user rating; each developer contains the information including name and the mashups he/she registered.

In order to examine the performance of our method, we separate the dataset into two sets: one set contains the mashups published from June 2005 to August 2012 which we use as the **Training Data** to calculate the global reputation and the topics in the ecosystem; the other set contains the mashups published from September 2012 to March 2013 which we use as the **Testing Data** to test the performance of our approach. Table 2 reports some basic statistics of our experiment dataset.

**Table 2.** Basic Statistic of the ProgrammableWeb Data for Experiment

| | **Training Period (2005.6~2012.8)** | **Testing Period (2012.9~2013.3)** |
|---|---|---|
| ***Number of Services*** | **7077** | **805** |
| ***Number of Mashups*** | ***6726*** | ***212*** |
| ***Number of Developer**** | **2383** | **127** |
| ***Number of Providers*** | **5905** | **699** |

*: Only the developers who publish at least one mashup are considered

## 4.2 Evaluation Metrics

In order to evaluate the performance of each approach, we use the existence of the services in the testing period as the ground truth.

$$y(Se_i) = \begin{cases} 1 & Se_i \text{ exist in the testing period} \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

As the invoked frequencies of different services during the testing period are different, we can use the frequency $f(Se_i)$ as the ground truth. If a service does not appear in the testing period, then $f(Se_i) = 0$. Based on these, we can consider the precision ( $P@k$ ), and discounted cumulative gain ( $D@k$ ) which are defined as:

$$P@k = \frac{\sum_{Se_i \in top-k} y(Se_i)}{k} \qquad D@k = \sum_{Se_i \in top-k} \frac{f(Se_i)}{\log(1+\pi(Se_i))} \tag{21}$$

Here $\pi(Se_i)$ is the position of the service in the reputation ranking list. Furthermore, given an approach as the baseline, we can calculate the difference discounted cumulative gain ( $DD@k$ ):

$$DD@k = \frac{D@k - D@k(Baseline)}{D@k(Baseline)} \tag{22}$$

## 4.3 Performance Comparison

**Global Reputation Ranking Based on URP Framework**

Based on the discussion shown above, we will consider three propagation models: 1) Top-Popularity Reputation Model (TPR) in which the reputation of each service is just based on its used frequency; 2) Page-Rank-Based Reputation Model (PRR) in which the reputation of each service is related to its co-occurrence services and 3) Developer-related Reputation Model (DRR) in which the reputation of the developers is taken into account. For each model we will consider the two initial strategies (EI and PI) we discuss in Section 2.4. Thus we can get six models such as: TPR+EI, TPR+PI, PRR+EI, PRR+PI, DRR+EI and DRR+PI.

In order to compare the performance of the heterogeneous network, we define two methods which just employ the information of the service co-occurrence network. In fact, the service co-occurrence network is a homogeneous network.

1) Top-Degree Reputation Model (TDR): The reputation of each service is the normalization of its network degree in the service co-occurrence network.

2) Homogeneous Page-Rank Reputation Model (HPRR): The page rank algorithm is run on the service co-occurrence network and then the reputation of each service is the normalization of its page rank value.

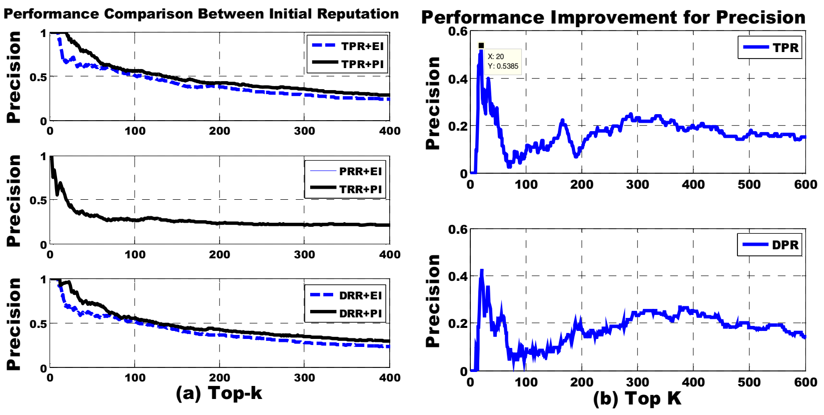

**Fig. 3.** Performance Comparison. (a) Performance comparison between different initial reputation of mashups. (b) Performance improvement in precision.

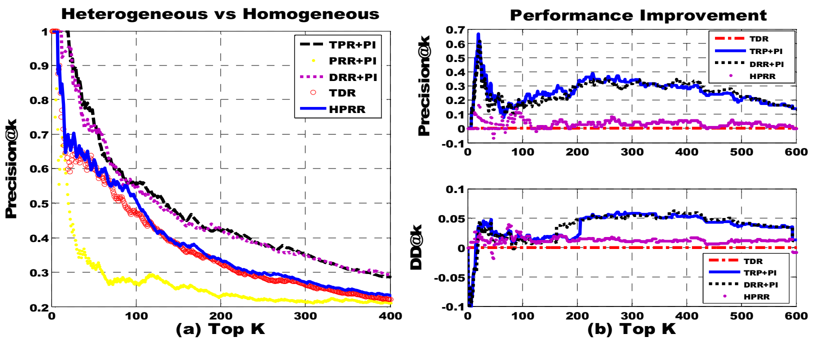

**Fig. 4.** Performance comparison between heterogeneous network methods and homogeneous network methods

Figure 3 shows the performance comparison between different initial reputations. We can observe that TPR model and DRR model are sensitive to the intitial mashup reputation. For TPR, the precision improvement for the top20 is up to 53.85% while for the long tail (top200~top500) is about 20%; For DRR, the precision improvement for top20 is up to 42.85% while for the long tail (top200~top500) is also about 20%. However for the PRR model, it is not sensitive to the initial mashup reputation and the two models have the extractly same performance. From the result, we can conclude that in practice, taking the mashup‘s popularity into account can help to improve the performance.

Figure 4 shows the performance comparison between the heterogeneous network and homogeneous network. From Figure 4 (a), we can observe that TPR+PI and DRR+PI gain consistently higher performance than TDR and HPRR. However, PRR+PI is the worst method in our experiment. One reason is that in ProgrammableWeb, the distribution of the service usage frequency meets the power-law distribution and only few services are very popular, for example, “*Google Map*“ appears in 2975 mashups. This makes the reputation propagation from service to mashup useless.

Here, we take the approach TDR as the baseline and Figure 4 (b) shows the performance improvement for the approaches except PRR+PI. We can observe that for the precision, HPRR gains a slight improvement; TRP+PI and DRR+PI gain the similar improvement, for top20 we gain a 66.67% improvement for TRP+PI and 61.53% for DRR+PI while for the long tail we gain a 30% improvement. Also the top15 DD@k for TRP+PI and DRR+PI is smaller than TDR while the precision for TRP+PI and DRR+PI are much better than TDR. This indicates that TRP+PI and DRR+PI will rank the services which are not that popular in a higher position. For example, in the top10, TRP+PI and DRR+PI will get the services such as “*Twilio*“, “*Twilio SMS*“, “*Foursquare*“ and “*Box.net*“ whose usage frequencies are not in the top10. However, the top10 services for TDR and HPRR are all the top10 popular services in the training period. Furthermore, for the long tail (top200~top500), we can observe that TRP+PI and DRR+PI gain a 5% improvement than TDR. Thus TRP+PI and DRR+PI can gain a higher performance than TDR and HPRR for the services which are not so popular in the past.

From the experiments shown above, we can conclude that: 1) The popularity-based reputation for mashups can improve the accuracy of the reputation ranking. 2) The heterogeneous network which contains richer information can gain a higher performance for the reputation ranking than the homogeneous network.

## Domain-Aware and Reputation-Aware Recommendation

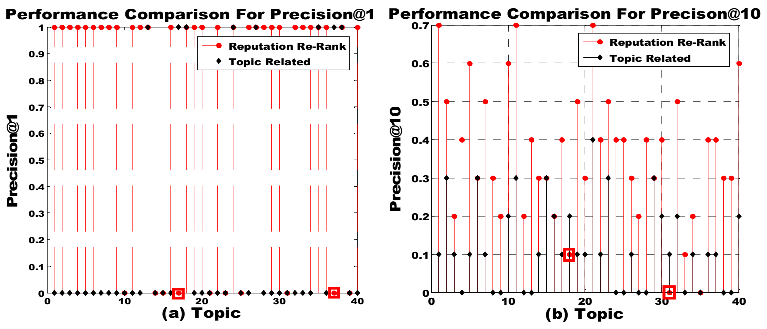

**Fig. 5.** Effectiveness of Reputation-based Recommendation

Another observation from the experiments is that the TRP+PI and DRR+PI gain higher performances than the others. Thus in this paper, we just employ DRR+PI to generate the global reputations for each services. Then the LDA-based topic model implemented in Mallet[4] is used to extract the different domains from the context of the services in the ecosystem. Here we set the number of domains to be 40. For each domain, we get the top 50 services based on the topic affiliate degree and consider then as the related services in the certain domain. Finally, we re-rank the services in each domain based on the global reputation and get the top-10 services as the recommendation for the providers. For simplicity, we name our approach as "RR".

In order to show the performance of our method, we consider the approach which only based on the domain affiliation degree. For simplicity, we name it as "TR". From Figure 5. We can observe that for the top1 recommendation, RR performances better than TR in 24 (60%) domains and worse than TR only in 2 (5%) domains; For the top10 recommendation, RR still performances better than TR in 60% domains and worse in 5% domains. Furthermore, the average improvement for the precison@10 is up to 118.54%.

## 5 Related Work

### 5.1 Reputation Based Trust for Recommendation

The concept of trust is not new. Trust has been studied in many disciplines including sociology, economics, and computer science [13]. In this paper, we consider the reputation based trust [14]. There are three groups of trust models for social networks: graph based trust models, interaction based trust models[7] and hybrid trust models [8]. These models aggregate the opinions of other users in the trust network to generate personalized recommendation for consumers. Recently, some researches employ the Quality of Service (Qos) combining with the Collaborative Filtering (CF) to calculate the reputation of the services for recommendation [6,8].

Recommender systems often exploit explicit trust data to generate recommended list. Explicit data sources include user profiles, articulated friend networks, or group memberships. Recommenders often exploit explicit friendships or linkages to generate recommendation lists [19, 21]. User profile data can be used to identify articles and other content believed to be relevant to users [22].

These approaches can yield good results when the services have complete metadata. However, in practice, most of the service ecosystem will not contain detail feedback from the users and the Qos for each service is resource-intensive to fetch, especially when considering the different Qos over time or at the different location. Thus from a different perspective, we just employ the historical popularity and the topological information to calculate the reputation of the services.

### 5.2 Complex Network for Service Ecosystem

The increasingly growth of Web services has attracted much attention in recent years. Many works employ the network analysis method to study the web service ecosystem as complex network analysis is a powerful tool to understand the large scale networks

[4] `http://mallet.cs.umass.edu/topics.php`

[15]. Yu and Woodard presented a preliminary result in studying the properties in ProgrammableWeb and proved that the cumulative frequency of APIs follows power law distribution [16]. Wang et al emphasized on mining mashup community from users' perspective by analyzing the User-API and User-Tag network in ProgrammableWeb [17]. Our previous work [18] studied the usage pattern of services in ProgrammableWeb based on the social network analysis.

Unlike the existing studies shown above, our work constructs a heterogeneous network for the service ecosystem and then formalizes the reputation propagation in the ecosystem. Furthermore, we take the difference between domains into account so that we can offer a domain-aware and reputation-aware recommendation.

## 6 Conclusion and Future Work

With the widely adoption of Service Oriented Architecture, we can observe a rapidly increasing number of services and their compositions these days. When exploring a service repository to choose services among those with similar functions, it is important to provide guidelines for the developers to select the trusted services. To the best of our knowledge, we are the first to: 1) Introduce a heterogeneous network model for a service ecosystem and the unified reputation propagation (URP) framework to calculate reputations in the service ecosystem. 2) Combine the LDA-based topic model with the URP to offer a domain-aware and reputation-aware service recommendation for the developers.

We conducted a comprehensive set of experiments on ProgrammableWeb and the results show the effectiveness of our method: 1) Taking the mashups' popularity into account gains a 40%~50% improvement for top20 precision and about 20% improvement for the long tail (top200~top500); 2) Compared with the method based on the homogeneous network, the heterogeneous network based methods such as TPR with PI and DRR with PI gain at least 60% improvement for top20 precision and 30% for the long tail (top200~top500); 3) Combining the reputation and domain, we get an 118.54% improvement for top10 precision compared to the domain-only method, which indicates that we can offer the trustworthy recommendation for each domain.

In the future, we will further our study on the implications of different parameters in our reputation propagation model, and the approach to find an optimal set of parameters for reputation calculation.

**Acknowledgments.** This work is partially supported by the National Natural Science Foundation of China (No. 61174169) and the National High Technology Research and Development Program of China (863Program, No. 2012AA040915)

## References

1. Al-Masri, E., Mahmoud, Q.H.: Investigating web services on the world wide web. In: Proc. 17th International Conference on World Wide Web, pp. 795–804 (2008)
2. Wang, J., Zhang, J., Hung, P.C.K., Li, Z., Liu, J., He, K.: Leveraging fragmental semantic data to enhance services discovery. In: IEEE International Conference on High Performance Computing and Communications (2011)
3. Blei, D.M., Lafferty, J.D.: Dynamic topic models. In: Proc. 23rd International Conference on Machine Learning, pp. 113–120 (2006)

4. Mollering, G.: The Nature of Trust: From Geog Simmel to a Theory of Expectation, Interpretation and Suspension. Sociology 35, 403–420 (2002)
5. Cook, K.S., Yamagishi, T., Cheshire, C., Cooper, R., Matsuda, M., Mashima, R.: Trust building via risk taking: A cross-societal experiment. Social Psychology Quarterly 68, 121–142 (2005)
6. Cao, J., Wu, Z., Wang, Y., Zhuang, Y.: Hybrid Collaborative Filtering algorithm for bidirectional Web service recommendation. Knowledge and Information Systems, 1–21 (2012)
7. Yao, J., Chen, S., Wang, C., Levy, D.: Modelling Collaborative Services for Business and QoS Compliance. In: Proc. International Conference on Web Services (ICWS), pp. 299–306 (2011)
8. Wang, Y., Vassileva, J.: A review on trust and reputation for web service selection. In: 27th International Conference on Distributed Computing Systems Workshops (2007)
9. Wu, Y., Yan, C., Ding, Z., Liu, G., Wang, P., Jiang, C., Zhou, M.: A Novel Method for Calculating Service Reputation. IEEE Transactions on Automation Science and Engineering 99, 1–9 (2013)
10. Zhang, J., Tan, W., Alexander, J., Foster, I., Madduri, R.: Recommend-As-You-Go: A Novel Approach Supporting Services-Oriented Scientific Workflow Reuse. In: IEEE International Conference on Services Computing, pp. 48–55 (2011)
11. Gupta, M., Sun, Y., Han, J.: Trust analysis with clustering. In: Proc. 20th International Conference Companion on World Wide Web, pp. 53–54 (2011)
12. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank citation ranking: bringing order to the web, Technical Report, Stanford Digital Library Technologies Project (1999)
13. Yao, J., Tan, W., Nepal, S., Chen, S., Zhang, J., De Roure, D., Goble, C.: ReputationNet: a Reputation Engine to Enhance ServiceMap by Recommending Trusted Services. In: IEEE Ninth International Conference on Services Computing, pp. 454–461 (2012)
14. Nepal, S., Malik, Z., Bouguettaya, A.: Reputation management for composite services in service-oriented systems. International Journal of Web Services Research 8, 29–52 (2011)
15. Tan, W., Zhang, J., Foster, I.: Network Analysis of Scientific Workflows: A Gateway to Reuse. IEEE Computer 43, 54–61 (2010)
16. Yu, S., Woodard, C.J.: Innovation in the programmable web: Characterizing the mashup ecosystem. In: Feuerlicht, G., Lamersdorf, W. (eds.) ICSOC 2008. LNCS, vol. 5472, pp. 136–147. Springer, Heidelberg (2009)
17. Wang, J., Chen, H., Zhang, Y.: Mining user behavior pattern in mashup community. In: IEEE International Conference on Information Reuse & Integration, pp. 126–131 (2009)
18. Huang, K., Fan, Y., Tan, W.: An Empirical Study of Programmable Web: A Network Analysis on a Service-Mashup System. In: IEEE 19th International Conference on Web Services, pp. 552–559 (2012)
19. Chen, J., Geyer, W., Dugan, C., Muller, M., Guy, I.: Make new friends, but keep the old: recommending people on social networking sites. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 201–210. ACM, Boston (2009)
20. Massa, P., Avesani, P.: Trust-aware recommender systems. In: Proc. ACM Conference on Recommender Systems (2007)
21. Guy, I., Ur, S., Ronen, I., Perer, A., Jacovi, M.: Do you want to know?: recommending strangers in the enterprise. In: Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work, pp. 285–294. ACM, Hangzhou (2011)
22. Liu, H., Maes, P.: Interestmap: Harvesting social network profiles for recommendations. Beyond Personalization-IUI (2005)