Incorporating Heterogeneous Information for Mashup Discovery with Consistent Regularization

Yao Wan[1], Liang Chen[2], Qi Yu[3], Tingting Liang[1], and Jian Wu[1]

[1] College of Computer Science & Technology, Zhejiang University, Hangzhou, China
{wanyao,liangtt,wujian2000}@zju.edu.cn
[2] School of Computer Science & Information Technology, RMIT, Melbourne, Australia, {liang.chen@rmit.edu.au}
[3] College of Computing & Information Sciences, Rochester Institute of Technology, Rochester, USA, {qi.yu@rit.edu}

Abstract. With the development of service oriented computing, web mashups which provide composite services are increasing rapidly in recent years, posing a challenge for the searching of appropriate mashups for a given query. To the best of our knowledge, most approaches on service discovery are mainly based on the semantic information of services, and the services are ranked by their QoS values. However, these methods can't be applied to mashup discovery seamlessly, since they merely rely on the description of mashups, but neglecting the information of service components. Besides, those semantic based techniques do not consider the compositive structure of mashups and their components. In this paper, we propose an efficient consistent regularization framework to enhance mashup discovery by leveraging heterogeneous information network between mashups and their components. Our model also integrates mashup discovery and ranking properly. Comprehensive experiments have been conducted on a real-world ProgrammableWeb.com[4] dataset with mashups and APIs[5]. Experimental results show that our model achieves a better performance compared with ProgrammableWeb.com search engine and a state-of-the-art semantic based model.

Keywords: mashup discovery, ranking, heterogeneous, regularization

1 Introduction

With the development of service oriented computing and the increasing demand of service consumers, a single web service is far from enough to satisfy the complex demand of users, contributing to the boom of composite services and mashups. More and more developers are using mashup technologies to build

[4] http://www.programmableweb.com
[5] In ProgrammableWeb.com, APIs are the service components of mashups. Our model verified on the ProgrammableWeb.com dataset could also be applied to other compositive service discovery scenarios.

their own services, and publishing them for other users to invoke. Statistics from ProgrammableWeb.com, a popular mashups and APIs management platform, show that the number of mashups has reached up to 6,094, and that the number of APIs has reached up to 10,634 by November 2014. In addition, the mashups are increasing on a daily basis. The rapid increase of mashups demands a systematic approach to discover mashups with great accuracy and efficiency.

There have been many research works focusing on service discovery and mashup discovery, which can be roughly divided into three categories: the traditional web service discovery, mashup discovery and mashup components discovery. Firstly, some literatures such as [9], [13] propose reasoning-based similarity algorithm to retrieve satisfied web services from the formalized description languages such as OWL-S and WSMO. But these methods cannot be applied to mashup discovery seamlessly for the reason that there is no formalized description for mashups. Secondly, in [10], [12], some semantic-based approaches are used for mashup discovery, while only the semantic information of mashups is considered, neglecting the relationship between mashups and their components. Thirdly, many research works [3], [11], [14] focus on discovery and recommendation for mashup components. Furthermore, ProgrammableWeb.com has its own mashup search engine, but how the search algorithm works is unknown to the public. According to our study, the mashup search system of ProgrammableWeb.com only supports weak semantics searching and gives low recall, losing many relevant results with similar semantics. For example, when searching "film", many mashups about "movie" will be lost, and when searching "Cellular phone", no results about "mobile" will be discovered.

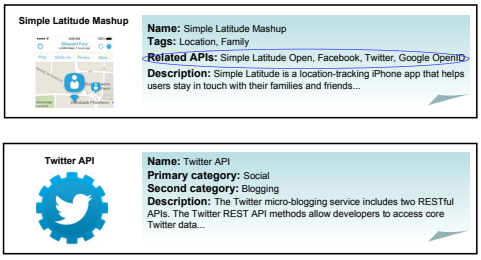


Fig. 1: An example of mashup and API

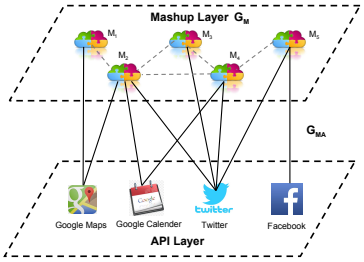


Fig. 2: A heterogeneous graph

In addition to the semantic information of mashups, the semantic information of their related components and the relationship between them should also be taken into consideration. Taking ProgrammableWeb.com as an example, it not only records the information of each mashup and API, but also the composition relationship between them. Figure 1 shows the detailed information of an mashup and API. Specifically, we can find that the tags, description of *Simple Latitude Mashup*, as well as APIs of which the mashup is composed from the website. Based on the information of Figure 1, a composition network between mashups

and their related APIs can be constructed. Links exist between mashups and APIs through the relation of "include" and "included by". Besides, in our paper we will also explore the relationship between mashups. If two mashups consist of a common API, there should be a link between them. These two networks containing the two described kinds of links are called heterogeneous network in our paper. A sample heterogeneous network of ProgrammableWeb.com is shown in Figure 2. In this figure, G_{MA} represents the network between mashups and APIs, G_M represents the network on the mashups layer.

The aim of this paper is to improve the discovery of mashups by introducing the semantic information of mashups as well their components, and leveraging the heterogeneous information network between mashups and their components. We first construct a heterogeneous network on mashups and APIs. A probabilistic model is proposed to calculate the relevance score of each mashup, integrating the semantic information of mashups and APIs, as well as the composition network between mashups and APIs. Furthermore, a regularized framework is proposed to ensure the consistency between mashups.

The contribution of this paper is summarized as follows:

- A probabilistic model is proposed to leverage the semantic information of mashups as well as their components. Besides, this model also integrate discovery and ranking process properly.
- A heterogeneous information network between mashups and their components is constructed and a regularized framework with consistency hypothesis is proposed to ensure the similarity consistency between mashups.
- We crawl 4,699 mashups and 937 related APIs from ProgrammableWeb.com to evaluate the performance of our approach. Comprehensive experiments show that our approach achieves a better performance comparing with ProgrammableWeb.com search engine and a baseline method.

The remainder of this paper is organized as follows. Section 2 introduces the related work of this paper. A probabilistic model and a heterogeneous network based mashup discovery approach are shown in Section 3. Section 4 describes the datasets we will use in our experiment and shows the experimental results and analysis. Finally, we conclude and give some future directions in Section 5.

2 Related Work

In service oriented computing, discovery of appropriate web services has always been a hot research topic. A number of approaches have been proposed for service discovery. Many semantic-based approaches develop reasoning-based similarity algorithms to retrieve relevant web services described using semantic web languages such as OWL-S and WSMO[9][13].

With the increase of mashup services, those traditional methods of service discovery cannot be applied seamlessly to mashup discovery, since most of them only consider the information extracted from WSDL documents. Recently, more and more research works are focusing on mashup searching or mashup discovery.

In [10], Li et al. provided a semantics extended framework to improve the precision and recall of mashup discovery as well as to improve the performance of the mashup discovery processing time. Elmeleegy et al. [8] presented a recommendation tool named *MashupAdvisor*, which used a semantic matching algorithm and a metric planner to modify the mashup to produce the suggested output. Bianchini et al. [2] proposed a recommendation system to design mashup applications based on the semantic description of mashup components, according to their similarities with designer's requirements and their mutual coupling.

Recently, some approaches using social networks to discover services are proposed. In [15], the authors combine current discovery techniques with social information as a mechanism to trade off exploration and exploitation. In [16], Zhou et al. provided an approach that learns a semantic Bayesian network with a semi-supervised learning method to build a web mashup network. Cao et al. [3] proposed a recommendation approach for mashup service that utilizes both users' interests and the social network based on relationships among mashups, APIs and tags. The inspiration of our paper mainly stems from some research on expertise finding. In [7], Deng et al. proposed a joint regularized framework to improve expertise retrieval by modeling heterogeneous networks as regularization constraints. In [18], an incremental method based on multiple graphs was proposed for document recommendation in a digital library. Besides, [17] provides a strong theoretical support for learning with local and global consistency. Inspired by those works, our paper builds a heterogeneous social network between mashups and their components, and a regularized framework is proposed to ensure the consistency.

3 Heterogeneous Network Based Mashup Discovery

3.1 Baseline Model for Mashup Discovery

The probabilistic model proposed in this subsection mainly follows the basic idea of a document-centric probatilistic model, which is proposed to estimate the expertise of a candidate by summing the relevance of its associated documents[7]. In the contex of mashup discovery, we denote the relevance score of candidate mashup m_i related to a given query q as $p(m_i|q)$, and according to the document-centric model,the relevance score of a mashup related to a given query can be formulated as:

$$\begin{aligned} p(m_i|q) &= \lambda \sum_{a \in \mathscr{A}_{m_i}} p(m_i|a)p(a|q) + (1-\lambda)p(m_i|q) \\ &\propto \lambda \sum_{a \in \mathscr{A}_{m_i}} p(m_i|a)p(q|a)p(a) + (1-\lambda)p(q|m_i)p(m_i) \end{aligned} \tag{1}$$

where $\mathscr{A}_{m_i}$ denotes the APIs set of which mashup m_i is composed of; $p(m_i|a)$ denotes the probability of mashup m_i being relevant to a given API a; $p(q|a)$ and $p(q|m_i)$ denote the semantic similarities of API and mashup, respectively, for a

given query; $p(a)$ and $p(m_i)$ can be seen as the quality of API a and mashup m_i, respectively; $\lambda(0 \leq \lambda \leq 1)$ is a tuning parameter used to determine how much the relevance score of a candidate mashup relies on the candidate mashup itself and its API components. In this equation, the Bayes' theorem is applied. Intuitively, we argue that when we calculate the relevance score of a mashup candidate, we should not only consider the information of the mashup, but also the information of its components.

The right hand of Eq.(1) can be divided into two parts. The first term represents the relevance score contributed by APIs by aggregating the relevance scores of APIs directly associated with a mashup. The second term denotes the relevance score from the mashup itself. Those two terms are combined by a tuning parameter λ. When $\lambda = 0$, only the information of mashup is considered, and when $\lambda = 1$, only the information of mashup's components is considered, otherwise, the semantic information of mashup as well as its components are integrated to improve the performance of mashup discovery.

Specifically, in this model, $p(m_i|a)$ represents the association between the candidate and its components. Suppose that mashup m_i consists of n_{m_i} APIs, then $p(m_i|a) = 1/n_{m_i}$ if a is a component of m_i, and zero otherwise. $p(q|a)$ measures the relevance between q and API a, while $p(q|m_i)$ measures the relevance between q mashup m_i. These two probabilities can be determined by using the language model. In this paper, we use Latent semantic indexing (LSI) mothod [6] to calculate the semantic similarities between a query and APIs or mashups. Before applying LSI, some standard process of natural language processing such as *case folding*, *tokenization*, *pruning*, *stemming* and *spell correcting* will be conducted on all APIs and mashups.

In addition, the prior probability $p(a)$ and $p(m)$ can be viewed as the quality of an API and mashup respectively, which generally follow the uniform distribution. Indeed, the quality of an API or mashup can also be set to be how popular the API or mashup is. In the context of our problem, since the popularity of mashups are difficult to measure, the qualities of mashups are set to be uniform. However, in the information network of APIs and mashups, we define the popularity of a particular API as the number of times that it is used in the formation of mashups. For example, in Figure 2, the *"Twitter API"* is used by four mashups, then the popularity of *"Twitter API"* is 4. From our analysis, we can find the distribution of API popularity is long tailed [1]. So we estimate $p(a)$ by the logrithm of the popularity of API.

For simplicity, let $\mathbf{x}$ be the relevance vector between API a_i and query q with $x_i = p(q|a_i)$, $\mathbf{y}$ be the relevance vector between mashup m_i and query q with $y_i = p(q|m_i)$, $\mathbf{Q}_A$ and $\mathbf{Q}_M$ be the diagonal matrix which represent the quality of APIs and mashups respectively, and $\mathbf{P}_{MA}$ be the composition matrix between mashups and APIs. The primary model as shown in Eq.(1) can be rewritten as:

$$\mathbf{z} = \lambda \mathbf{P}_{MA} \mathbf{Q}_A \mathbf{x} + (1 - \lambda) \mathbf{Q}_M \mathbf{y} \tag{2}$$

where $\mathbf{z}$ represents the relevance score vector of all candidate mashups, and λ is the tuning parameter which controls the weight between the mashups and their components.

3.2 Heterogeneous Information Incorporation

In the previous subsection, we utilize the description of mashups and their components to measure the relevance of candidate mashups for a given query. In addition to the textual document information, some information of the heterogeneous network should also be considered. In this subsection, we will describe a mashup consistency hypothesis and enforce the hypotheses by defining regularization constraints.

Mashup Consistency Hypothesis: *If two mashups share many common services with respect to a given query, then their relevance score in the queried field should be similar in some sense.* As shown in Figure 2, mashup m_2 is composed of *"Google Maps API"*, *"Google Calendar API"* and *"Twitter API"*, while mashup m_4 is composed of *"Twitter API"* and *"Google Calendar API"*, these two mashups share two common APIs, so we can consider that their functionality and quality are similar in some sense, and their relevance for a given query should also be similar.

According to [17], we enforce the above hypothesis by defining the regularization constraints. Suppose we are given a mashup graph $G_M = (V_M, E_M)$, which is a weighted undirected graph. Suppose that the pairwise similarities among the mashups are described by matrix $S_M \in \mathbb{R}^{|M|\times|M|}$ measured based on G_M. Thus, we formulate to minimize a regularization loss function as follows:

$$\Omega(\mathbf{z}) = \mathbf{z}^T(\mathbf{I} - \mathbf{S}_M)\mathbf{z} + \mu \left\|\mathbf{z} - \mathbf{z}^0\right\|^2 \tag{3}$$

$$s.t. \quad \mathbf{z}^0 = \lambda \mathbf{P}_{MA}\mathbf{Q}_A\mathbf{x} + (1-\lambda)\mathbf{Q}_M\mathbf{y} \tag{4}$$

where $\mu > 0$ is the regularization parameter. The first term of the loss function defines the *mashup consistency*, which prefers small difference in relevance scores between nearby mashups; the second term is the *fitting constraint* that measures the difference between final scores $\mathbf{z}$ and the initial relevance scores $\mathbf{z}^0$. The initial relevance score vector $\mathbf{z}^0$ can be calculated according to Eq.(2) in the probabilistic model. Setting $\partial\Omega(\mathbf{z})/\partial\mathbf{z} = 0$, we can see that the solution $\mathbf{z}^*$ is essentially the solution to the linear equation:

$$(\mathbf{I} - \alpha\mathbf{S}_M)\mathbf{z}^* = (1-\alpha)\mathbf{z}^0 \tag{5}$$

where $\alpha = 1/(1+\mu)$. Since the matrix $\mathbf{S}_M$ is usually very sparse, calculating the inversion of $\mathbf{S}_M$ is of high time complexity. One solution to the above equation is using a powerful iterative method [17]:

$$\mathbf{z}^{t+1} \leftarrow \beta\mathbf{S}_M\mathbf{z}^t + (1-\beta)\left[\lambda\mathbf{P}_{MA}\mathbf{Q}_A\mathbf{x} + (1-\lambda)\mathbf{Q}_M\mathbf{y}\right] \tag{6}$$

where $\beta = 1/(1+\mu)$, $\mathbf{z}^* = \mathbf{z}^\infty$ is the solution.

Now the interesting question is how to calculate $\mathbf{S}_M$ among set M. For graph data, a number of works [4] have been given on obtaining the similarity measures. For undirected graph, $\mathbf{S}_M$ is simply the normalized adjacency matrix $\mathbf{W}$:

$$\mathbf{S}_M = \mathbf{\Pi}^{-1/2}\mathbf{W}\mathbf{\Pi}^{1/2} \tag{7}$$

where $\mathbf{W}$ is the adjacency matrix of mashups in G_M, $\mathbf{W}_{ij} = 1$ if node i is linked to node j, otherwise, $\mathbf{W}_{ij} = 0$, and $\mathbf{\Pi}$ is a diagonal matrix with $\mathbf{\Pi}_{ii} = \sum_j \mathbf{W}_{ij}$.

3.3 Implementation

Figure 3 shows the framework of the proposed heterogeneous information network based mashup discovery, which integrates the semantic information of mashups and their components, as well as the similarity consistency between mashups. The proposed framework is comprised of three components: *heterogeneous network construction*, *data processing*, and *mashup ranking*. The construction of heterogeneous network and data processing can be performed offline, while the mashup ranking part should be conducted online according to the query specified by a user.

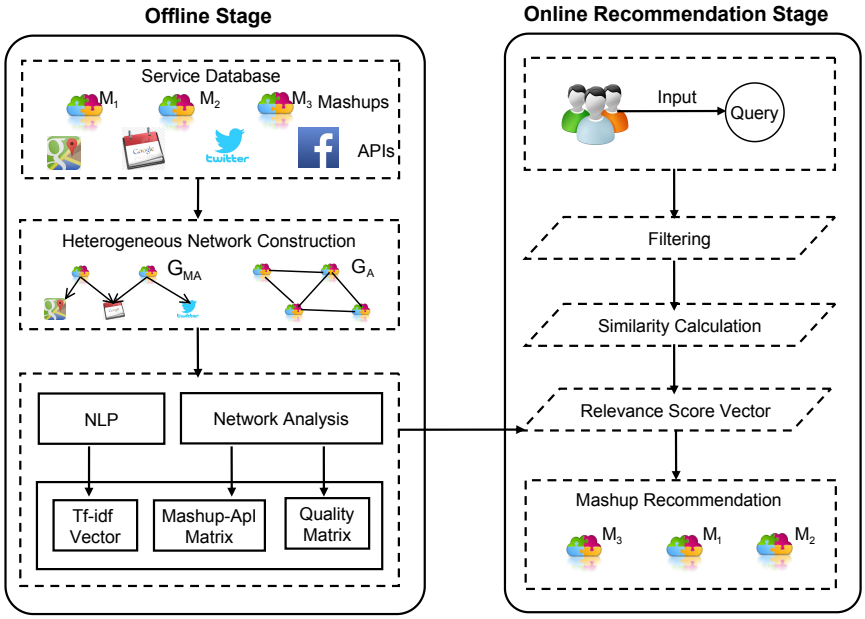


Fig. 3: The Heterogeneous Network Based Mashup Discovery Framework

4 Experiments

4.1 Experimental Setup

Dataset ProgrammableWeb.com is one of the most popular platforms that has collected lots of APIs and mashups used in Web and mobile applications. To evaluate our proposed approach, we crawl all the mashups and their related APIs from ProgrammableWeb.com as we can.

From the statistics, we totally get 4,699 mashups and 937 APIs in our data collection. After the construction of the heterogeneous graph, we observe that there are many edges on the mashup graph, while relatively few edges in the graph of mashups and APIs. As for G_M the density of matrix is nearly 17.0%, while for G_A the density of matrix is only 0.18%.

Evaluation Metrics For the evaluation, several categories of Web search evaluation metrics are used to measure the performance of our proposed model from different aspects, including some relevance based metrics, ranking based metrics and diversity based metrics. To measure the relevance of our search results, we use the precision at rank k ($P@K$) which is widely used and is defined as: $P@K = \frac{\#\ relevant\ in\ top\ K\ results}{K}$. $P@K$ measures the fraction of the top-K retrieved results that are relevant for the given query. From the ranking aspect, we use Mean Reciprocal Rank (MRR) to evaluate the ranking of our search results. A larger MRR value means a better result. The MRR is defined as: $MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i}$, where $|Q|$ is the size of query set. We expect that our search results should not only have a high precision and reasonable ranking, but also have a high diversity. Following [5], we use the α-DCG metric to measure the novelty and diversity of our retrieved results. The α-DCG is defined as: $\alpha\text{-}DCG_K = \sum_{i=1}^{K}\frac{G_i}{log_2(i+1)}$, where $G_i = \sum_{j=1}^{n} J(m_i,j)(1-\alpha)^{\sum_{k=1}^{i-1} J(m_k,j)}$; n is the total number of topics the searching results contains; $J(d_i,j) = 0$, if result m_i contains topic j, otherwise, $J(m_i,j) = 1$. $\sum_{k=1}^{i-1} J(m_k,j)$ represents the degree of diversity and novelty of the searching result; α is a parameter. In our model, α is set to be 0.5.

4.2 Comparison

In this subsection, comparisons between our method and the following approaches have been made to show the effectiveness of our proposed approach.

- **PW Search Engine**: The ProgrammableWeb.com has its own search engine for mashups and APIs discovery. From our observation, we find that the search results is mostly based on the name, description and tags/categories of mashups/APIs. In addition, the search results are all ranked by their updated date.
- **MD-Sim**: This method which has been used in many state-of-the-art works just utilizes the semantic information of mashups. The semantic similarities between mashups and query are calculated by the LSI method.
- **MD-Sim+**: Compared with the above two methods, our probabilistic model (MD-Sim+) employs the semantic information of mashups, as well as the semantic information of the related APIs which are the components of mashups. The qualities of related APIs are also introduced in this approach.
- **MD-HIN**: This method is the extended version of the probabilistic model (MD-Sim+). It proposes a consistency hypothesis on mashups firstly, and a regularization constraints is employed.

Before comparing the performance of the above four methods, several points should be made clear first. Since there is no published ground truth for comparison, we select twenty queries of different topics to evaluate the performance on the above metrics. According to the search results, if the categories of mashup are related to a query, then the mashup will be considered to be relevant with the query. We judge the degree of relevance by the number of followers of mashups. Since most mashups have more than one tags, we will use the tags to evaluate the diversity of results. The parameter settings of our approaches are $\lambda = 0.4$, $\beta = 0.5$, $\#iteration = 100$, and topic number of LSI is set as 20. The experimental results are shown in Table 1, and the detailed investigations of parameter settings will be provided in Section 4.3 and Section 4.4.

	P@10	P@20	P@50	MRR	α-DCG
PW	**0.595**	0.493	0.431	-	-
MD-Sim	0.555	0.488	0.553	0.121	2.920
MD-Sim+	0.575	0.495	0.534	0.137	2.916
(vs **MD-Sim**)	+3.60%	+1.43%	-3.44%	+13.22%	-0.01%
MD-HIN	0.555	**0.53**	**0.537**	**0.160**	**3.027**
(vs **PW**)	-6.72%	+7.51%	+24.59%	-	-
(vs **MD-Sim**)	0.00%	+8.61%	-2.89%	+32.23%	+3.66%

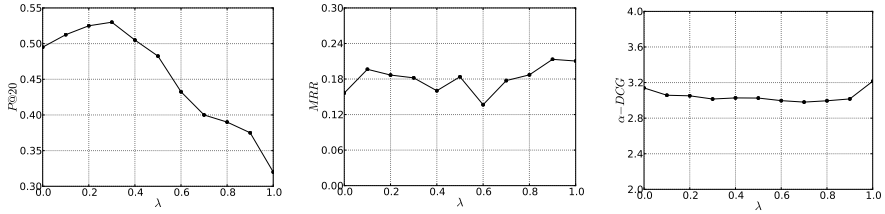
Table 1: Experimental results of our proposed method and other methods. The percentages of relative improvements(%) are also shown in this table.

Based on the results in Table 1, we have the following observations:

- From the perspective of $P@K$, when the value of K is small, the performance of ProgrammableWeb.com search engine is a litter better than our approach (MD-HIN). While when the value of K is large, our approach has a great advantage over ProgrammableWeb search engine. This is because as K increases, the ProgrammableWeb.com search engine is unable to discover so many mashups, which just depends on the utilization of the mashup information, while our approach can find more related mashups by incorporating the mashups and APIs' information and leveraging the heterogeneous social network between them.
- From the perspective of ranking of results, our extented approach (MD-HIN) achieves better performance than the probabilistic approach (MD-SIM) since introducing the quality of APIs along with the similarity consistency on mashup social network, making sure that the mashups with similar quality will have similar ranking scores.
- Among all the discovery methods, our proposed method (MD-HIN) generally achieves better performance on both $P@20$, MRR and α-DCG, indicating that integrating the semantic information of mashups with APIs, and considering the similarity consistency on mashup social network will facilitate and improve the discovery of mashups. These experimental results demonstrate

that our model leveraging the heterogeneous social network is practical and effective.

4.3 Impact of λ



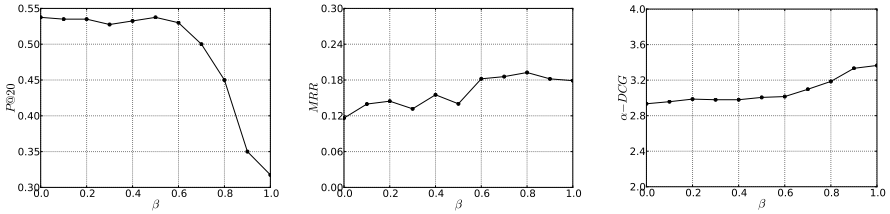
(a) Impact of λ on P@20 (b) Impact of λ on MRR (c) Impact of λ on α-DCG

Fig. 4: Impact of λ

In our model, the parameter λ controls how much our method relies on the semantic information of mashups and their related APIs. To study the impact of λ on P@20, MRR and α-DCG, we vary λ from 0 to 1 with a step value 0.1. The experimental results are shown in Figure 4. Figure 4(a) shows that optimal λ value settings can achieve better performance of mashup discovery, which demonstrates that fusing the information of mashups and their related APIs with our proposed approach will improve the discovery accuracy. As λ increases, the P@20 value increases at first, but when λ surpasses a certain threshold, the P@20 value decreases with further increase of the value of λ. This phenomenon confirms the intuition that purely using the semantic information of mashups or purely employing the semantic information of their related APIs cannot generate better performance than fusing these two factors together. From Figure 4(b) and Figure 4(c), we can find that the value of λ also has an impact on the MRR and α-DCG although the impact is small. This demonstrates that when we introduce the semantic information and quality of APIs, the ranking and diversity of search results will be improved.

4.4 Impact of β

In our model, $\beta = 1/(\mu+1)$ where μ is a regularization parameter which controls the difference between final score $\mathbf{z}$ and the initial score $\mathbf{z}^0$. To study the impact of β on the metrics of our approach, we change β from 0 to 1 with a step value 0.1. We set $\lambda = 0.4$, Top-K=20, and $\#iteration = 100$ in this experiment. From Figure 5(a), we can find that value β has a significant impact on the precision of the discovery results. When β increases to 1, the P@20 will decrease rapidly. This is because that when β is near to 1, μ is near to 0, in this condition, we only consider the similarity constraints of mashups, neglecting the constraint



(a) Impact of β on $P@20$ (b) Impact of β on MRR (c) Impact of β on α-DCG

Fig. 5: Impact of β

that the final score value should be fitting to the initial value. Figure 5(b) and Figure 5(c) show that the value of β still has an effect on MRR and α-DCG, although the effect is not obvious.

5 Conclusion and Future Work

Based on some traditional semantic-based service discovery methods, we propose an approach to improve mashup discovery by integrating the semantic information of mashups and their related APIs. Besides, a similarity consistency between mashups is proposed and a regularization framework is employed to achieve better performance. Comprehensive experiments on a real-world ProgrammableWeb.com dataset are conducted, and the extensive experimental analysis shows the effectiveness of our approach.

Although the data crawled from ProgrammableWeb.com is sufficient for evaluation purpose, we believe that there is a necessity for an experiment on traditional Web services described by a standard language such as WSDL. In our future work, we plan to build a social network on traditional Web services and verify our model. In addition, we will extend our ground truth with more queries, and more information retrieval evaluation metrics will be introduced to enhance the quality of our work. Furthermore, we are going to conduct more research on the social network, a more complex social network including service users will be built.

Acknowledgments

This research was partially supported by the Natural Science Foundation of China under grant of No. 61379119, Science and Technology Program of Zhejiang Province under grant of No. 2013C01073, the Open Project of Qihoo360 under grant of No. 15-124002-002.

References

1. Anderson, C.: The long tail: how endless choice is creating unlimited demand. Random House (2007)

2. Bianchini, D., De Antonellis, V., Melchiori, M.: A recommendation system for semantic mashup design. In: Database and Expert Systems Applications (DEXA), 2010 Workshop on. pp. 159–163. IEEE (2010)
3. Cao, B., Liu, J., Tang, M., Zheng, Z., Wang, G.: Mashup service recommendation based on user interest and social network. In: Web Services (ICWS), 2013 IEEE 20th International Conference on. pp. 99–106. IEEE (2013)
4. Chung, F.R.: Spectral graph theory, vol. 92. American Mathematical Soc. (1997)
5. Clarke, C.L., Kolla, M., Cormack, G.V., Vechtomova, O., Ashkan, A., Büttcher, S., MacKinnon, I.: Novelty and diversity in information retrieval evaluation. In: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval. pp. 659–666. ACM (2008)
6. Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W., Harshman, R.A.: Indexing by latent semantic analysis. JAsIs 41(6), 391–407 (1990)
7. Deng, H., Han, J., Lyu, M.R., King, I.: Modeling and exploiting heterogeneous bibliographic networks for expertise ranking. In: Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries. pp. 71–80. ACM (2012)
8. Elmeleegy, H., Ivan, A., Akkiraju, R., Goodwin, R.: Mashup advisor: A recommendation tool for mashup development. In: Web Services, 2008. ICWS'08. IEEE International Conference on. pp. 337–344. IEEE (2008)
9. Klusch, M., Fries, B., Sycara, K.: Automated semantic web service discovery with owls-mx. In: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems. pp. 915–922. ACM (2006)
10. Li, C., CHENG, B., CHEN, J., LI, C., WANG, G., GU, P., LI, D.: A semantics extended indexes framework for mashup discovery. Journal of Computational Information Systems 7(5), 1446–1454 (2011)
11. Ni, Y., Fan, Y., Huang, K., Bi, J., Tan, W.: Negative-connection-aware tag-based association mining and service recommendation. In: Service-Oriented Computing, pp. 419–428. Springer (2014)
12. Riabov, A.V., Boillet, E., Feblowitz, M.D., Liu, Z., Ranganathan, A.: Wishful search: interactive composition of data mashups. In: Proceedings of the 17th international conference on World Wide Web. pp. 775–784. ACM (2008)
13. Rohallah, B., Ramdane, M., Zaidi, S.: Agents and owl-s based semantic web service discovery with user preference support. arXiv preprint arXiv:1306.1478 (2013)
14. Tapia, B., Torres, R., Astudillo, H.: Simplifying mashup component selection with a combined similarity-and social-based technique. In: Proceedings of the 5th International Workshop on Web APIs and Service Mashups. p. 8. ACM (2011)
15. Torres, R., Tapia, B., Astudillo, H.: Improving web api discovery by leveraging social information. In: Web Services (ICWS), 2011 IEEE International Conference on. pp. 744–745. IEEE (2011)
16. Zhou, C., Chen, H., Peng, Z., Ni, Y., Xie, G.: A semantic bayesian network for web mashup network construction. In: Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom). pp. 645–652. IEEE (2010)
17. Zhou, D., Bousquet, O., Lal, T.N., Weston, J., Schölkopf, B.: Learning with local and global consistency. Advances in neural information processing systems 16(16), 321–328 (2004)
18. Zhou, D., Zhu, S., Yu, K., Song, X., Tseng, B.L., Zha, H., Giles, C.L.: Learning multiple graphs for document recommendations. In: Proceedings of the 17th international conference on World Wide Web. pp. 141–150. ACM (2008)