Service Package Recommendation for Mashup Development Based on a Multi-level Relational Network

Jian Cao(✉), Yijing Lu, and Nengjun Zhu

Department of Computer Science and Engineering, Shanghai Jiaotong University, Shanghai, People's Republic of China
{cao-jian,luyjcathy,zhu_nj}@sjtu.edu.cn

Abstract. With the number of services growing explosively, it has been a serious problem selecting appropriate services for mashup development. In this paper, we come up with a Multi-level Relational Network (MRN) based approach for service recommendation in mashup development, which captures deep relationships among services on top of latent topic, tag and service network. Specifically, by modeling the correlation among services, representing it as a Quadratic Knapsack Problem and solving it using Branch and Bound algorithm, we are able to recommend a package of services, which are complementary and possible to be used together in a mashup. Experiments on a realistic mashup data set have shown its effectiveness.

Keywords: Mashup creation · Multi-level relational network · Service package recommendation

1 Introduction

With the advent of Web 2.0, mashup, as a novel service composition implementation, is developing rapidly. There are some websites or communities providing supports for mashup, such as Yahoo Pipes[1], ProgrammableWeb[2], etc. Despite the convenience to create a mashup, there are still troubles finding suitable services to compose, because of the large amount of services available. As is often the case, users type down a few words describing the functionality of their mashups to be developed, and then get stuck on which service to choose. Therefore, recommending services in mashup development has been a vital problem. There is already some reasearch aiming to solve this. Information retrieval technology can be used to match the descriptions of mashup and APIs. It is intuitive but ignores the knowledge that can be captured from existing mashups. Another common approach is to discover frequent service composition sequences from historical mashups. In this case, when some services are chosen, other services can

[1] pipes.yahoo.com.
[2] www.programmableweb.com.

Q.Z. Sheng et al. (Eds.): ICSOC 2016, LNCS 9936, pp. 666–674, 2016.
DOI: 10.1007/978-3-319-46295-0_46

be recommended by searching relating frequent service composition sequences. However, in this case, the newly published services have no chance to be recommended. Moreover, these two approaches can only recommend optional services with similar function. Obviously, in mashup development, it is more useful to recommend a set of compatible services, rather than a list of similar services.

In this paper, we call the problem of recommending a set of compatible services for mashup development "service package recommendation". The problem we intend to solve is how to recommend a service package when a textual description for the mashup to be developed is given.

Our work is distinguished by three key contributions: (1) We propose a MRN model, which can capture the deep relationships among services and support service recommendation. (2) We formalize the service package recommendation as a Quadratic Knapsack Problem and solve it using Branch and Bound algorithm. (3) We carry out experiments on realistic data set which show that our approach is effective.

The remaining sections are organized as follows: Sect. 2 describes related work. In Sect. 3, we present our MRN model. In Sect. 4, we formalize the service package recommendation problem and describe in detail how it can be solved. The experiments on realistic dataset are presented in Sect. 5. Finally, we draw a conclusion of this paper in Sect. 6.

2 Related Work

One major service recommendation approach takes advantage of historical frequent service composition patterns for recommendation [1,2]. However, simply relying on frequent sequence fails in the situations where new patterns can be adopted, or when new services are published.

Since users are searching for some APIs to use, adopting traditional information retrieval technologies is also an intuitive method [3]. Furthermore, Chune Li et al. [4] adopted relational topic model in mashup development, which take into account the historical associations between mashup and API. Similar to topic model, tag as a user-driven way to feature mashup and APIs, is also capable of annotating frequent patterns [5]. These research gives us the inspiration that semantic approaches, though not working well singly, are helpful if properly used.

Some researchers are also investigating the possibility of applying recommender system and approaches in e-commerce into mashup domain [6–8] . But in our problem where only description is provided, these approaches cannot be applied directly.

Furthermore, there is some research trying to make use of more information to improve the recommendation. Among those approaches, the most widely used information is social relationship [9,10]. However, such social-aware methods rely too much on a complete social network. Once social information is incomplete or absent, they are not going to work well.

Our MRN model integrates topics, tags and services and their inherent relationships in a more comprehensive way. In addition, in the hope of recommending services with complementary functionality, we get inspiration from [11] which implemented bundle recommendation (similar to package recommendation) in e-commerce.

3 The Multi-level Relational Network Model

Our approach is based on the MRN model, which involves the relationships of topics, tags and services. In the rest of this section, we introduce each of the layers in detail.

3.1 Service Network

The bottom layer of the MRN is the service network. Once two services appear in the same mashup, we record them as a service pair (s_i, s_j). And thus, we can construct a service network, whose edge weight (i, j) indicates how many times (s_i, s_j) are connected based on historical mashup information.

To use service network, we assign a score between the given mashup and each of the candidate APIs upon service layer. The score is denoted as service utility. In the rating step, for mashup m_i and service s_j, the service utility is constructed as follows: we first obtain a list of services RS_{m_i} relevant to m_i through topic model, and then accumulate the edge weights between services on top of service network.

3.2 Tag Network

For a mashup m_i, the tags it has are denoted as $Tag_{m_i} = \{tag_{m_i,1}, tag_{m_i,2}, \cdots, tag_{m_i,j}, \cdots\}$. Services s_i also have their tags, denoted as $Tag_{s_i} = \{tag_{s_i,1}, tag_{s_i,2}, tag_{s_i,3}, \cdots, tag_{s_i,j}, \cdots\}$. We apply the association rule mining method described in [5] to discover the relationships between tags. For each mashup m_i, we consider Tag_{m_i} to be related to all the tags belonging to S_{m_i}. And also, since all the services in S_{m_i} are composed together, each pair of them is considered being associated in the tag network.

By identifying all the tag pairs in the mashup repository, we can get a network $Tag\_network$, in which the weight of each edge (i, j) represents how many times tag_i and tag_j are associated.

Given a mashup and a candidate set of APIs, we can assign a utility score between the mashup and each API from the viewpoint of tag network. In another word, we try to describe which API is closer to the mashup in a quantity way based on tag network. This can be done by taking out the tags of mashup and API, and accumulating the weight of the edge between each tag pair.

3.3 Topic Network

It is very useful to apply topic model in recommendation. LDA is used to model our resources. For each mashup and API, we collect their descriptions as corpus and extract a topic distribution. A topic distribution for mashup m_i is denoted as $Topic_{m_i} = \{topic_{m_i,1}, topic_{m_i,2}, topic_{m_i,3}, \cdots, topic_{m_i,\#TN}\}$, in which $\#TN$ is a number manually defined. Top K topics in the distribution are chosen to be relevant topics.

We use topic model as a part of our rating metrics, in the similar way we use tags. A topic pair $(topic_i, topic_j)$ is considered associated if they belong to a mashup and its API respectively, or to two APIs belonging to the same mashup. Finally, we can construct a topic network, the weight of each edge (i, j) represents how many times $topic_i$ and $topic_j$ are connected.

Moreover, like tag utility, we assign a score called topic utility to measure the cooperation degree between a mashup and API connection from the viewpoint of topic network, by cross-multiplying the relevant topics and adding their scores over topic network.

4 Service Package Recommendation

Package recommendation is sometimes called bundle recommendation, especially in e-commerce. Bundle originally refers to a set of items that customers consider or buy together [11]. When recommending a list of services to users, we want them to have the highest utilities, not only to the mashup, but also internally to each other. Thus, we can say that we are not recommending a list of services, but rather a package of services.

4.1 Model Construction

We propose our new model to calculate the total utilities. Given a mashup m_D and a set of services, we use a vector x to denote which service we pick into the final result. x_i is 0 if s_i is not chosen, and 1 if chosen. Then, the total utility can be denoted as:

$$U_{total}(m_D, x) = \sum_i U(m_D, s_i)x_i + \sum_{i<j} U(s_i, s_j)x_i x_j \tag{1}$$

The goal of service package recommendation is to find a set of k items that maximize the total utility. Let:

$$r_i = U(m_D, s_i)$$

$$Q_{i,j} = \begin{cases} U(s_i, s_j) & \textbf{if} i \neq j \\ 0 & \textbf{if} i = j \end{cases}$$

The total utility can now be described as:

$$U_{total}(x) = r^T x + x^T Q x \tag{2}$$

It is essentially a case of Quadratic Knapsack Problem [12].

$$\max_{x\in\{0,1\}^n,|x|=k} r^T x + x^T Q x \tag{3}$$

4.2 Solving QKP

Having $r = 0$, it is not hard to see that this problem reduces to a k-clique problem. Since k-clique problem is a NP-hard problem, it is supported by theorem that our problem is also a large-scale NP-hard problem. Luckily, we get the inspiration from [11] that solving the problem for all the items is equivalent to solving it for a carefully constructed candidate set.

As said by Zhu et al., it is observed in practice that only a few items have a high score, so it is intuitively understandable to construct a candidate set. As to our data set, we also observed such situation that to a given description of mashup, only a small set of services have relatively high utilities, while the majority have a low value. It is rigorously proved in [11] that items dominated by k or more items will not appear in an optimal selection, in which dominance is a relationship between items. Item α dominants item β if:

$$r_\alpha + Q_{\alpha\alpha} + \min_{A\subseteq U,|A|=k-1} \sum_{i\in A, i\neq\alpha} (Q_{\alpha i} + Q_{i\alpha}) >$$
$$r_\beta + Q_{\beta\beta} + \max_{B\subseteq U,|B|=k-1} \sum_{i\in B, i\neq\beta} (Q_{\beta i} + Q_{i\beta})$$

That implies we only need to find items dominated by no more than $k-1$ items, which makes the candidate set size different for every item. Computations and empirical results show that the resultant candidate size is very small compared to the whole dataset. To decrease computational time and cost, it is acceptable to choose a fixed candidate set size m for every item, which is slightly larger than empirical data. So in our model, it is sufficient to solve the problem on top of a candidate set C_{set}.

We update our model as follows, where r_c and Q_c are the responding variables for C_{set}. With the size of potential candidates largely reduced, our problem is now tractable.

$$\max_{x\in\{0,1\}^n,|x|=k} r_c^T x + x^T Q_c x \tag{4}$$

We use Gurobi[3] as an assistance, which is a state-of-art mathematical programming solver.

5 Experiments

5.1 Dataset and Compareative Methods

The dataset we use comes from ProgrammableWeb.com, namely the largest mashup information sharing community. We crawled ProgrammableWeb and get

[3] http://www.gurobi.com/.

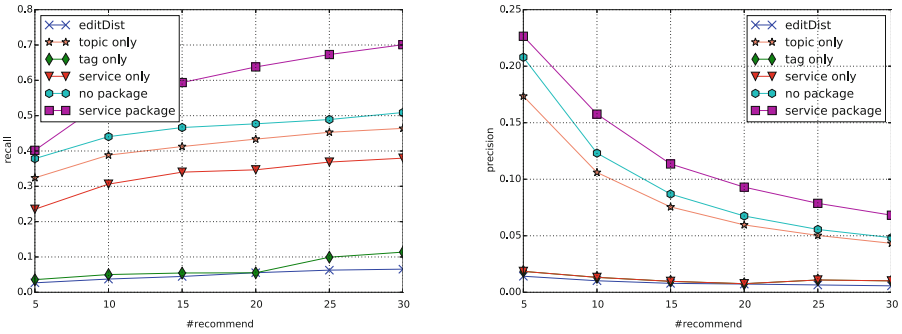
7674 valid mashups as well as 10240 services (called APIs in ProgrammableWeb). For each mashup and service, we got its full information especially service lists for mashup, description and annotating tags for both of them. There are totally 462 different tags involved. We selected information retrieval, recommendations based on single layer network and recommendation without considering package effect as our comparative methods.

5.2 Model Training

We use 90 % of the mashups to learn the model, and the rest 10 % to test. There are many parameters in our model. In this section, we show how they are selected and fitted.

The first parameter to deal with is the number of topics. We use the Stanford Topic Modeling Toolbox[4] to help train our topic model. We collect the descriptions of mashups and services together to train topic model with number of topics varying from 10 to 400. The best performance is reached when topic number equals 300, which is later chosen for our further experiments.

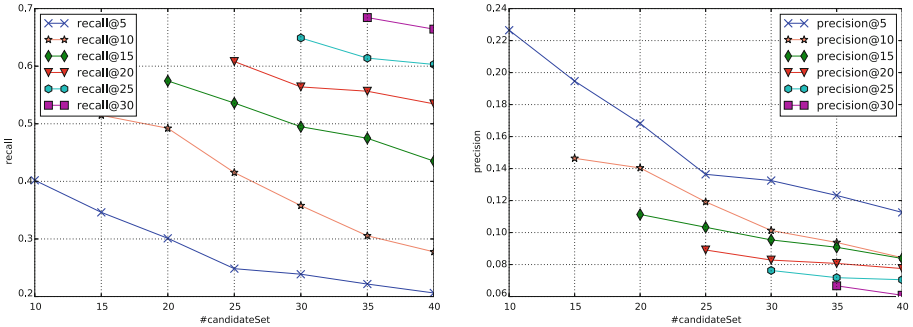
Another important arguments in our model is the weight of each network layer. It is hard to decide how much a layer should contribute to the final result theoretically. We can try different combination of values under some constraints, and get a best fit. To achieve this goal, we employ genetic algorithm to find the answer under a linear constraints. Finally we get the best fit of (α, β, γ) at $(0.4, 0.1, 0.5)$ respectively for topic network, tag network, and service network. This confirms the leading position of service layer in the recommendation process, but others can also help improve the results.



(a) Recall (b) Precision

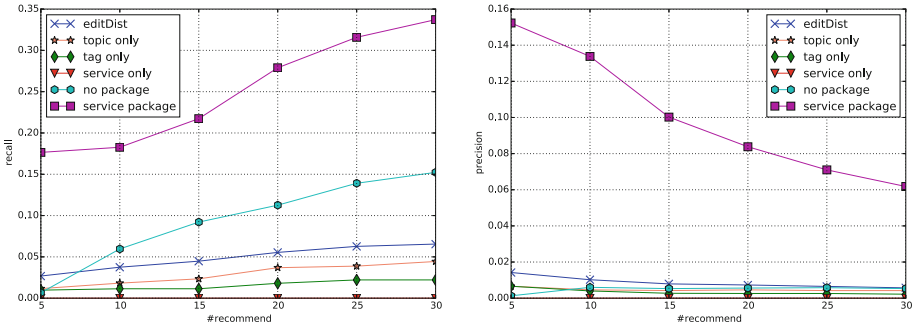
Fig. 1. Results of different approaches over various #recommendation

[4] http://www.nlp.stanford.edu/software/tmt/tmt-0.4/.



(a) Recall (b) Precision

Fig. 2. Results of service package recommendation over various #candidateSet and #recommendation



(a) Recall (b) Precision

Fig. 3. Results when recommended services are new

5.3 Results and Discussion

Upon the selected best parameter values, we conducted experiments using each of the methods. Figure 1 shows the recall for all the methods over varying recommendation size.

It can be seen in the figure that, the methods of textual similarity through EditDistance performs worst, as expected, since it ignores many useful historical composition information. The three separate base approaches comes next, with topic method performing the best. The model based on MRN outperforms the comparative approaches to a great extent. Moreover, our service package recommendation approach further improves the performance.

The service package recommendation result in Fig. 1 is a part of our final experiments. We show the full results in Fig. 2, where various length of candidate set is considered, as well as the size of recommendation set. It is unexpected to see that given a recommendation number, the smaller candidate set size is,

the better performance we get. This suggests that we need to choose a proper candidate set in practice.

It has been mentioned that our model is capable of recommending new services. We conduct an experiment to verify this. For each test case, we remove the effect of its services in the train set to make the services all new. The results are shown in Fig. 3. It can be seen from the figure that the integrated model can successfully recommend new services.

6 Conclusion and Future Work

In this paper, a multi-level relational network is applied to capture and model the comprehensive relationships between topics, tags, and services. Specifically, we propose the concept of service package recommendation, enabling to recommend a set of complementary services, rather than a list of similar ones. Experiments on realistic mashup data set have shown its effectiveness.

Acknowledgements. This work is partially supported by China NSF (Granted Number 61272438,61472253), Research Funds of Science and Technology Commission of Shanghai Municipality (Granted Number 15411952502).

References

1. Oliveira, F.T., Murta, L., Werner, C., Mattoso, M.: Using provenance to improve workflow design. In: Freire, J., Koop, D., Moreau, L. (eds.) IPAW 2008. LNCS, pp. 136–143. Springer, Heidelberg (2008). doi:10.1007/978-3-540-89965-5_15
2. Maaradji, A., Hacid, H., Skraba, R., Vakali, A.: Social web mashups full completion via frequent sequence mining. In: Proceedings - 2011 IEEE World Congress on Services, SERVICES 2011, pp. 9–16 (2011)
3. Platzer, C., Dustdar, S.: A vector space search engine for Web services. In: Third European Conference on Web Services (ECOWS 2005) (2005)
4. Li, C., Zhang, R., Huai, J., Sun, H.: A novel approach for API recommendation in mashup development. In: 2014 IEEE International Conference on Web Services (ICWS) (2014)
5. Goarany, K., Kulczycki, G., Blake, M.B.: Mining social tags to predict mashup patterns. In: Proceedings of the 2nd International Workshop on Search and Mining User-Generated Contents (SMUC 2010), pp. 71–78. ACM, New York (2010)
6. Cremonesi, P., Picozzi, M., Matera, M.: A comparison of recommender systems for mashup composition. In: Proceedings of 2012 3rd International Workshop on Recommendation Systems for Software Engineering, RSSE 2012, pp. 54–58 (2012)
7. Yao, L., Wang, X., Sheng, Q.Z., Ruan, W., Zhang, W.: Service recommendation for mashup composition with implicit correlation regularization. In: 2015 IEEE International Conference on Web Services (ICWS), pp. 217–224, June 2015
8. Zheng, Z., Lyu, M.R.: Component recommendation for cloud applications. In: Proceedings of RSSE, pp. 48–49 (2010)
9. Cao, B., Liu, J., Tang, M., Zheng, Z., Wang, G.: Mashup service recommendation based on user interest and social network. In: 2013 IEEE 20th International Conference on Web Services (ICWS), pp. 99–106 (2013)

10. Xu, W., Cao, J., Hu, L., Wang, J., Li, M.: A social-aware service recommendation approach for mashup creation. In: 2013 IEEE 20th International Conference on Web Services (ICWS), pp. 107–114 (2013)
11. Zhu, T., Harrington, P., Li, J., Tang, L.: Bundle recommendation in ecommerce. In: Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2014, pp. 657–666 (2014)
12. Gallo, G., Hammer, P., Simeone, B.: Quadratic knapsack problems. In: Padberg, M.W. (ed.) Combinatorial Optimization, vol. 12, pp. 132–149. Springer, Heidelberg (1980)