

Multi-relation Based Manifold Ranking Algorithm for API Recommendation

Fenfang Xie^(✉), Jianxun Liu, Mingdong Tang, Dong Zhou,
Buqing Cao, and Min Shi

Key Laboratory of Knowledge Processing and Networked Manufacturing,
Hunan University of Science and Technology, Xiangtan 411201, China
xiefragrance@gmail.com, ljx529@gmail.com,
tangmingdong@gmail.com, zdbloom@gmail.com,
buqingcao@gmail.com, toshimin132@gmail.com

Abstract. The number of APIs on the Web has increased rapidly in recent years. It becomes quite popular for developers to combine different APIs to build innovative Mashup applications. However, it is challenging to discover the appropriate ones from enormous APIs for Mashup developers (i.e., API users). In order to recommend a set of APIs that most satisfy the users' requirements, we propose a multi-relation based manifold ranking approach. The approach exploits the textual descriptions of existing Mashups and APIs, as well as their composition relationships. It firstly groups Mashups into different clusters according to their textual descriptions, then explores multiple relations between Mashup clusters and between APIs. Finally, it employs a manifold ranking algorithm to recommend appropriate APIs to the user. Experiments on a real-world dataset crawled from ProgrammableWeb.com validate the effectiveness of the proposed approach.

Keywords: Mashup clustering · Web API · Multi-relation · Manifold ranking · API recommendation

1 Introduction

Mashup, as an emerging Web development mode, enables a developer easily combining the content from more than one source on the Web to create new applications for end-users. For example, a developer can combine the addresses and photographs of their library branches with a Google map to create a map Mashup [1]. With the spread of service computing and cloud computing, the past decade has witnesses a tremendous growth in the Application Programming Interfaces (APIs) published on the Web, which provide a variety of services such as data, storage, computing, and communication. How to Mashup APIs to create new applications thus has attracted great attention from both industry and academia. Although a user can occasionally employ a single API to meet his/her needs in Mashup creation, more often than not, the fulfilment of his/her needs relies on a combination of APIs. Therefore, to develop a Mashup, identifying a set of related APIs rather than a single specific one, becomes a key task.

According to latest statistics, the number of APIs and Mashup applications has been growing exponentially in recent years [2]. This observation indicates that finding

appropriate APIs for Mashup developers become increasingly difficult, even for an experienced developer. For one thing, most APIs only provide a single functionality, which may not satisfy users' comprehensive needs. Therefore, APIs are often composed together to build a single application. How to discover and choose the appropriate APIs to compose for users' requirement is critical [3]. For another, only a few APIs were ever used by the users and the reuse rate of most services is rather low [4]. How to improve the usage rate of the existing APIs is also an important issue.

In order to solve the two problems mentioned above, we propose a multi-relation based manifold ranking algorithm to assist Mashup developers discovering a list of suitable APIs. Firstly, we cluster existing Mashups into groups according to their textual descriptions so that the Mashups within a cluster are similar in functionality. When a user proposes his/her requirement, we can measure the similarities between the user's requirement and the Mashup clusters to identify which cluster matches the best with the user's requirement. This can save more time than matching the user's requirement with the description of every single Mashup. Secondly, we measure the similarity relations between Mashup clusters by exploiting cosine similarity according to TF-IDF (Term Frequency–Inverse Document Frequency) vectors of them so that the Mashup clusters can associate with each other in a way of similar functionality. Thirdly, we measure the similarity relation, composition relation and potential composition relation between existing APIs so that a set of APIs relevant to the user's requirement can be identified. Finally, by integrating the different relations and the popularity of APIs, we employ a manifold ranking algorithm to recommend top- K most appropriate APIs to the user. Our approach can not only recommend APIs that are popular in the Mashups, but also recommend APIs that have similarity relations, composition relations, and potential composition relations with each other.

The main contributions of this paper are outlined below:

- We propose a multi-relation based manifold ranking algorithm to recommend APIs for Mashup creation according to the user's requirement.
- We define several relations between Mashups and between APIs, and present a set of algorithms to mine the relations.
- We conduct a set of experiments on a real-world dataset, experimental results validate the effectiveness of the proposed approach and show that our approach outperforms baseline approaches.

The remainder of the paper is organized as follows: Sect. 2 presents the framework of our approach. Section 3 gives a detailed description of our multi-relation based manifold ranking algorithm for API recommendation. Section 4 describes the experiments and discusses the results. Section 5 surveys the related work. Section 6 concludes the paper.

2 Framework

In this section, we describe the framework of our approach. The framework contains two parts: offline part and online part, which is shown in Fig. 1. The offline part includes the following main steps:

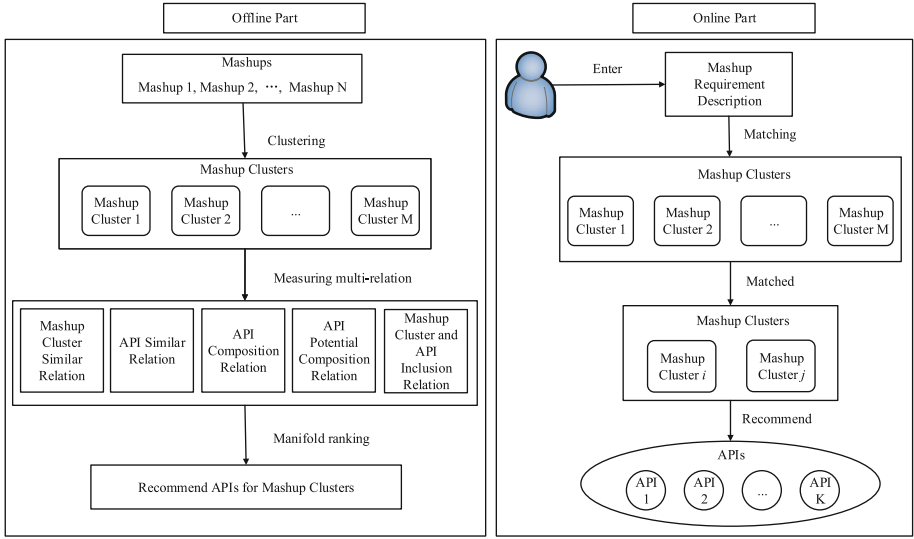


Fig. 1. The API recommendation framework of our approach

- Cluster Mashups into groups so that Mashups within a cluster are closely connected based on the similarities of their textual descriptions.
- Measure the similarity relations between Mashup clusters by using TF-IDF vectors generated from their textual descriptions.
- Measure the similarity relation, composition relation, and potential composition relation between APIs.
- Measure the inclusion relation between Mashup clusters and APIs according to the number of times that an API included in a Mashup cluster.
- Aggregate the above similarity relation, composition relation, potential composition relation and apply a manifold ranking algorithm to recommend APIs for every Mashup clusters.

The online part includes four main steps:

- Input requirement for Mashup creation. The user specifies his/her requirement of functions that he/she would like to develop for a Mashup creation.
- Match the input requirement with Mashup clusters. We calculate the similarities between the user's requirement description and every single Mashup cluster.
- Identify the matched clusters. On the basis of the similarities between Mashup clusters and the user's requirement, we choose two most similar clusters.
- Recommend APIs for Mashup creation. After identifying the similar Mashup clusters, we recommend the user with those APIs that are not only popular in the Mashups, but also have similarity relation, composition relation and potential composition relation with each other.

It is worth noting that we choose two most similar Mashup clusters instead of one single cluster in our approach. This is because the user's requirement may be

complicated and needs more than one API to implement. Only one Mashup cluster may not be able to fulfill the user's requirement. For example, if a Mashup developer wants to create a Mashup that has the functions of location and sending messages, both the map cluster and the message cluster of Mashups may be needed. One can also choose three or more similar Mashup clusters to cover even more APIs. However, this may increase the algorithm complexity and reduce the precision of API recommendation.

3 Approach

In this section, we first describe the Mashup clustering algorithm, then introduce how to mine the relations between Mashup clusters and between APIs. At last, we present our proposed multi-relation based manifold ranking algorithm for API recommendation.

3.1 Mashup Clustering

Firstly, we collect the textual information of all Mashups. Secondly, we preprocess the Mashup description data, such as filtering stop words; extracting stem of words, and so on. Thirdly, we convert the preprocessed textual description of each Mashup into a TF-IDF vector. Finally, we use the *K-Medoids* algorithm [5] to cluster Mashups into similar groups in functionality. Specially, in order to measure the similarity relations between Mashup clusters, each Mashup cluster is also represented by a TF-IDF vector, which is computed by simply aggregating the description of each Mashup in it.

K-Medoids is a classical partitioning technique of clustering that clusters the data set of n objects into k clusters known a priori. A medoid can be defined as the object of a cluster whose average dissimilarity to all the objects in the cluster is minimal, i.e. it is a most centrally located point in the cluster. It is more robust to noises and outliers as compared to *K-Means* because it minimizes a sum of pair-wise dissimilarities instead of a sum of squared Euclidean distances [6].

3.2 Relation Definition and Mining

In this section, we define several relations: similarity relation between Mashup clusters and between APIs, composition & potential composition relations between APIs, and inclusion relation between Mashup clusters and APIs. We measure the similarity relations between Mashup clusters so that the similar Mashup clusters can be associated with each other. We measure the similarity, composition and potential composition relations between existing APIs so that a set of APIs relevant to a user's requirement can be identified. We measure the inclusion relation between Mashup clusters and APIs so that it will be conducive to recommend APIs that are popular in the Mashups.

Definition 1 (Similarity relation): Both the similarity relation between Mashup clusters and between APIs are exploited in this work. Let m be the number of Mashup clusters, n be the number of APIs. Let M be an $m \times m$ matrix representing the similarity relation between Mashup clusters. Let A^{sim} be an $n \times n$ matrix representing the similarity relation between APIs. Let V_i and V_j be the TF-IDF vectors of Mashup clusters or APIs.

The cosine similarity of two vectors can be calculated by using the following formula:

$$Sim(v_i, v_j) = \frac{V_i \cdot V_j}{|V_i| \cdot |V_j|} \quad (1)$$

Thus, $M_{ij} = Sim(m_i, m_j)$ and $A_{ij}^{sim} = Sim(a_i, a_j)$.

Definition 2 (Composition relation): The composition relation between two APIs represents that the two APIs have been jointly used by at least one Mashup, i.e., have been composed for creating at least one Mashup. Let A^{com} be an $n \times n$ matrix representing the composition relation between APIs. $\Psi(a_i)$ be a set of Mashups that invokes API a_i . We employed the idea of resource allocation [7, 8] to measure the composition relation in this paper.

Given a pair of API a_i and a_j , which can be used by different Mashups, among these Mashups, some Mashups may invoke both of the APIs. These Mashups can be regarded as resources allocated to a_i and a_j . If a_i and a_j have common Mashups, they are considered to have a composition relation between them. In this regard, the composition relation between API a_i and a_j can be calculated with the following formula:

$$\ell(a_i, a_j) = \sum_{c \in \Psi(a_i) \cap \Psi(a_j)} \frac{1}{k(c)} \quad (2)$$

where c is the common Mashup that invokes both a_i and a_j , and $k(c)$ is the number of APIs that c has used.

The values of $\ell(a_i, a_j)$ are likely to be greater than 1. So, we normalize the values of $\ell(a_i, a_j)$ into range [0, 1] using the following formula:

$$Com(a_i, a_j) = 1 - e^{-\ell(a_i, a_j)} \quad (3)$$

where $Com(a_i, a_j)$ represents the normalized composition relation, $A_{ij}^{com} = Com(a_i, a_j)$. Obviously, the larger of $\ell(a_i, a_j)$, the larger $Com(a_i, a_j)$ will be, i.e., the stronger is the composition relation between API a_i and a_j .

Definition 3 (Potential Composition relation): The composition relation between two APIs means that the two APIs have not been composed for Mashup creation, but have potential to be composed. Let A^{pcom} be an $n \times n$ matrix representing the potential composition relation between APIs. Based on the previously calculated similarity relation and composition relation, we measure the potential composition relation between APIs through a link prediction ideology.

To infer potential composition relation, we develop two heuristic rules based on our observations:

- Heuristics 1: If API a_i has a composition relation with API a_k , and API a_k has a composition relation with API a_j , then a_i and a_j are likely to be composed for Mashup creation. To verify this statement, we analysed the API network based on the composition relationship and found that the clustering coefficient of the network

is 0.618, which is quite high, indicating there are many triangle-like structures in the network. Therefore, Heuristics 1 is reasonable.

- Heuristics 2: If API a_i has a composition relation with API a_k , and API a_k is very similar (or equivalent) to API a_j , then a_i and a_j are likely to be composed for Mashup creation. Because API a_k and API a_j are very similar, a_j can be considered as an alternative of a_k . This Heuristics is also reasonable due to our intuition.

The potential composition relations between APIs based on Heuristics 1 can be formulated as:

$$P^{H1}com(a_i, a_j) = (A^{com}A^{com})_{ij} = \sum_{k \in \Gamma(a_i) \cap \Gamma(a_j)} Com(a_i, a_k) \times Com(a_k, a_j) \quad (4)$$

where $\Gamma(a_i)$ and $\Gamma(a_j)$ are neighbor set of API a_i and a_j respectively. The more APIs acting as the role of a_k , the greater is the value of $P^{H1}com(a_i, a_j)$.

In a similar manner, to calculate the potential composition relation between APIs based on Heuristics 2, we adopt the following formula:

$$P^{H2}com(a_i, a_j) = (A^{com}A^{sim})_{ij} = \sum_{k \in \Gamma(a_i) \cap \Gamma(a_j)} Com(a_i, a_k) \times Sim(a_k, a_j) \quad (5)$$

Finally, we combine both Heuristics 1 and Heuristics 2 to infer the potential composition relation between APIs. The computation formula is:

$$Pcom(a_i, a_j) = \rho P^{H2}com(a_i, a_j) + (1 - \rho) P^{H1}com(a_i, a_j) \quad (6)$$

where $Pcom(a_i, a_j)$ represents the final potential composition relation between APIs a_i and a_j , and ρ is a real number in range $[0,1]$ which can be customized according to specific application scenarios, $A_{ij}^{pcom} = Pcom(a_i, a_j)$.

Definition 4 (Inclusion relation): The inclusion relation between a Mashup cluster and an API means that the API has been used by at least one Mashup in the Mashup cluster. We use an $m \times n$ matrix P to represent the inclusion relation between Mashup clusters and APIs, where each entry denotes the number of times an API was included in a Mashup cluster. We can also model the inclusion relations between Mashup clusters and APIs as a heterogeneous bipartite graph.

3.3 Multi-relation Based Manifold Ranking Algorithm

Manifold ranking, a semi-supervised graph based ranking algorithm, has been widely applied in information retrieval (such as image retrieval), and shown to have excellent performance and feasibility on a variety of data types. The core idea of manifold ranking is to rank the data with respect to the intrinsic structure collectively revealed by a large number of data [9–11].

1. Original manifold ranking algorithm description

Given a set of data $X = \{x_1, x_2, \dots, x_n\} \subset R^m$ and build a graph on the data. Let $W \in R^{n \times n}$ be the adjacency matrix wherein each entry w_{ij} represents the weight of the edge between point i and j . Let $F : X \rightarrow R$ be a ranking function which assigns to each point x_i via a ranking score F_i . Finally, we define an initial vector $q = [q_1, \dots, q_n]^T$, in which $q_i = 1$ if x_i is a query and $q_i = 0$ otherwise.

The cost function associated with F is defined to be

$$O(F) = \frac{1}{2} \left(\sum_{i,j=1}^n w_{ij} \left\| \frac{1}{\sqrt{D_{ii}}} F_i - \frac{1}{\sqrt{D_{jj}}} F_j \right\|^2 + v \sum_{i=1}^n \|F_i - q_i\|^2 \right) \quad (7)$$

where $v > 0$ is the regularization parameter and D is a diagonal matrix with $D_{ii} = \sum_{j=1}^n w_{ij}$. The first term in the cost function is a smoothness constraint, which makes the nearby points in the space have close ranking scores. The second term is a fitting constraint, which means the ranking result should fit to the initial label assignment.

2. Our manifold ranking algorithm

In our application scenario, let $M_R = [M_{R1}, \dots, M_{Rm}]^T$ and $A_R = [A_{R1}, \dots, A_{Rn}]^T$ be ranking results of Mashup clusters and APIs respectively. Since we are trying to rank APIs for a given Mashup, we set $M_Q = [M_{Q1}, \dots, M_{Qm}]^T$ in which query Mashup clusters are set to 1 and others are set to 0, and $A_Q = [A_{Q1}, \dots, A_{Qn}]^T$ in which the corresponding APIs for the query Mashup cluster are set to 1.

After obtained the above five matrices, namely, $M, A^{sim}, A^{com}, A^{pcom}, P$, we employ our multi-relation based manifold ranking algorithm to recommend APIs.

By integrating multiple relations (i.e. the similarity relation between Mashup clusters and between APIs, the composition and potential composition relations between APIs, and the inclusion relation between Mashup clusters and APIs) into the original manifold ranking algorithm, our multi-relation based manifold ranking algorithm consists of seven terms, the first five terms are smoothness constraints, and the last two terms are fitting constraints. Details are as follows:

- The similarity relation between Mashup clusters

$$\chi_1 = \frac{1}{2} \alpha \sum_{i,j=1}^m M_{ij} \left(\frac{1}{\sqrt{D_{M_{ii}}}} M_{Ri} - \frac{1}{\sqrt{D_{M_{jj}}}} M_{Rj} \right)^2 \quad (8)$$

This term makes the similar Mashup clusters have close ranking scores.

- The similarity relation between APIs

$$\chi_2 = \frac{1}{2} \beta \sum_{i,j=1}^n A_{ij}^{sim} \left(\frac{1}{\sqrt{D_{A_{ii}^{sim}}}} A_{Ri} - \frac{1}{\sqrt{D_{A_{jj}^{sim}}}} A_{Rj} \right)^2 \quad (9)$$

This term makes the similar APIs have close ranking scores.

- The composition relation between APIs

$$\chi_3 = \frac{1}{2} \theta \sum_{i,j=1}^n A_{ij}^{com} \left(\frac{1}{\sqrt{D_{A_{ii}^{com}}}} A_{Ri} - \frac{1}{\sqrt{D_{A_{jj}^{com}}}} A_{Rj} \right)^2 \quad (10)$$

This term makes the composable APIs have close ranking scores.

- The potential composition relation between APIs

$$\chi_4 = \frac{1}{2} \lambda \sum_{i,j=1}^n A_{ij}^{pcom} \left(\frac{1}{\sqrt{D_{A_{ii}^{pcom}}}} A_{Ri} - \frac{1}{\sqrt{D_{A_{jj}^{pcom}}}} A_{Rj} \right)^2 \quad (11)$$

This term makes the potential composable APIs have close ranking scores.

- The inclusion relations between Mashup clusters and APIs

$$\chi_5 = \gamma \sum_{i=1}^m \sum_{j=1}^n P_{ij} \left(\frac{1}{\sqrt{D_{PM_{ii}}}} M_{Ri} - \frac{1}{\sqrt{D_{PA_{jj}}}} A_{Rj} \right)^2 \quad (12)$$

This term makes the popular APIs in a certain Mashup cluster have close ranking scores.

- The fitting constraint

$$\chi_6 = \mu \sum_{i=1}^m (M_{Ri} - M_{Qi})^2 + \eta \sum_{i=1}^n (A_{Ri} - A_{Qi})^2 \quad (13)$$

These two terms make the APIs' ranking results be consistent with the queried Mashup cluster.

In the above seven terms, $0 < \alpha, \beta, \theta, \lambda, \gamma, \mu, \eta < 1$ are the regularization parameters and we set $\alpha + \beta + \theta + \lambda + \gamma + \mu + \eta = 1$. Wherein, α controls the similarity relation between Mashup clusters, β controls the similarity relation between APIs, θ controls the composition relation between APIs, λ controls the potential composition relation between APIs, γ controls the inclusion relation between Mashup clusters and APIs.

Based on the above seven terms, we model our objective function as follows:

$$\Phi(M_R, A_R) = \sum_{t=1}^6 \chi_t \quad (14)$$

Our goal is to minimize the objective function and infer A_R from $M, A^{sim}, A^{com}, A^{pcom}, P, M_Q, A_Q$. By minimizing it, we can get the ranking of Mashup clusters and APIs as close as possible to the given training data.

In order to simplify the objection function, we symmetrically normalize $M, A^{sim}, A^{com}, A^{pcom}, P$ by the form like $S = D^{-1/2} W D^{-1/2}$ [12]. With simple derivations, each part can be transformed to vector representation form as $M_R^T(I - S_M)M_R, A_R^T(I - S_A^{sim})A_R, A_R^T(I - S_A^{com})A_R, A_R^T(I - S_A^{pcom})A_R, M_R^T M_R + A_R^T A_R - 2M_R^T S_P A_R$ [13].

Then we can rewrite the objective function in the equivalent matrix-vector form:

$$\begin{aligned} \Phi(M_R, A_R) = & \alpha M_R^T(I - S_M)M_R + \beta A_R^T(I - S_A^{sim})A_R + \theta A_R^T(I - S_A^{com})A_R + \lambda A_R^T(I - S_A^{pcom})A_R \\ & + \gamma(M_R^T M_R + A_R^T A_R - 2M_R^T S_P A_R) + \mu(M_R - M_Q)^T(M_R - M_Q) + \eta(A_R - A_Q)^T(A_R - A_Q) \end{aligned} \quad (15)$$

Where I is the corresponding identity matrix.

We minimize the objective function with respect to M_R and A_R by differentiating it and set the corresponding derivatives to 0. Namely,

$$\frac{\partial \Phi}{\partial M_R} = [(1 - \beta - \theta - \lambda - \eta)I - \alpha S_M]M_R - \gamma S_R A_R - \mu M_Q = 0 \quad (16)$$

$$\frac{\partial \Phi}{\partial A_R} = [(1 - \alpha - \mu)I - \beta S_A^{sim} - \theta S_A^{com} - \lambda S_A^{pcom}]A_R - \gamma S_P^T M_R - \eta A_Q = 0 \quad (17)$$

Let $\varphi = (1 - \alpha - \mu)I - \beta S_A^{sim} - \theta S_A^{com} - \lambda S_A^{pcom}$ and $\phi = (1 - \beta - \theta - \lambda - \eta)I - \alpha S_M$, then we can obtain A_R .

$$A_R = \varphi^{-1} \left[\gamma S_P^T (\phi - \gamma^2 S_P \varphi^{-1} S_P^T)^{-1} (\mu M_Q + \gamma S_P \varphi^{-1} \eta A_Q) + \eta A_Q \right] \quad (18)$$

After computing A_R , we can employ it to recommend APIs for mashup clusters.

3. Recommend APIs for Mashup creation

Through our multi-relation based manifold ranking algorithm, we can obtain the recommended APIs for Mashup clusters. Now we can recommend APIs for Mashup developers according to his/her requirement specification. Firstly, we calculate the cosine similarities between the user's requirement and the Mashup clusters to identify which clusters match best with the user's requirement. Secondly, we choose two most similar Mashup clusters, because the developer may have a variety of requirements. Thirdly, after matching the two Mashup cluster, we recommend top- K APIs for Mashup creation. What's more, we can recommend APIs are not only popular in the

mashups, but also have similarity relation, composition relation, and potential composition relation with each other.

4. General expression

For the ease of understanding, we use pseudo code to describe our multi-relation based manifold ranking algorithm in Table 1. Line 00 clusters Mashups into groups according to their textual description and aggregates the Mashup descriptions of each clusters as the whole description of the Mashup cluster; Lines 01–05 compute the similarity relation between Mashup clusters; Lines 06–11 compute the similarity and composition relations between APIs; Lines 15–23 compute the potential composition relation between APIs; Line 25 builds the bipartite graph between Mashup clusters and APIs which is based on the number of times APIs are invoked by Mashups in a certain Mashup cluster; Line 26 integrates M , A^{sim} , A^{com} , A^{pcom} , P , M_Q , A_Q and employs the manifold ranking algorithm to recommend APIs for Mashup clusters; Line 27 obtains the top- K APIs recommendation list for Mashup clusters.

Table 1. The multi-relation based manifold ranking algorithm

Algorithm: multi-relation based manifold ranking
Input: A set of APIs a_1, a_2, \dots, a_n ; a set of Mashups M_1, M_2, \dots, M_n ; the textual descriptions of Mashups and APIs; the user's requirement for Mashup creation.
Output: A set of APIs to a Mashup user's specific requirement description.
00: Cluster all Mashups into groups according to their textual description
01: For each Mashup cluster m_i do
02: For each Mashup cluster m_j do
03: $Sim(m_i, m_j) \leftarrow$ similarity degree between m_i and m_j
04: End for
05: End for
06: For each API a_i do
07: For each API a_j do
08: $Sim(a_i, a_j) \leftarrow$ similarity degree between a_i and a_j
09: $Com(a_i, a_j) \leftarrow$ composition ability between a_i and a_j
10: End for
11: End for
12: $M_{ij} \leftarrow$ store $Sim(m_i, m_j)$ to similarity relation matrix
13: $A_{ij}^{sim} \leftarrow$ store $Sim(a_i, a_j)$ to similarity relation matrix
14: $A_{ij}^{com} \leftarrow$ store $Com(a_i, a_j)$ to composition relation matrix
15: For each element A_{ij}^{com} do
16: For each element A_{ij}^{com} do
17: $P^{H1}com(a_i, a_j) \leftarrow$ multiply A_{ij}^{com} by A_{ij}^{com}
18: End for
19: For each element A_{ij}^{sim} do
20: $P^{H2}com(a_i, a_j) \leftarrow$ multiply A_{ij}^{com} by A_{ij}^{sim}
21: End for
22: End for
23: $Pcom(a_i, a_j) \leftarrow$ combine $P^{H1}com(a_i, a_j)$ and $P^{H2}com(a_i, a_j)$
24: $A_{ij}^{pcom} \leftarrow$ store $Pcom(a_i, a_j)$ to potential composition relation matrix
25: Build the inclusion relation P between mashup clusters and APIs according to the popularity of APIs
26: Assemble M , A^{sim} , A^{com} , A^{pcom} , P , M_Q , A_Q to recommend APIs for Mashup clusters by using (18).
27: For each Mashup cluster save top- K ranking APIs in A_R

4 Experiment

In this section, we present a set of experiments to validate our approach on a real dataset and give an analysis of experimental results.

4.1 Dataset Description

In our experiments, the dataset used is crawled from ProgrammableWeb.com website in the range from June 2005 to December 2013. We remove Mashups that are of the same name and contain no more information except its name. After manually pre-processing the dataset, we obtain a collection of 5955 Mashups and 1069 APIs. An overview of the APIs information and Mashups information are shown in Tables 2 and 3 respectively. Moreover, an overview of the dataset is shown in Table 4. We use this dataset to recommend APIs for Mashup developers. All of the experiments are conducted on computer with Intel(R) Core(TM) i3 CPU(3.2 GHz and 6.0 GB RAM) and all algorithms are implemented in Matlab 2014.

Table 2. The information of APIs

APIID	APIName	APIDescription
1	Cloudmade Leaflet	CloudMade provides application developers with tools and APIs for creating unique location based applications across all major web and mobile platforms. Leaflet is a modern, lightweight BSD-licensed JavaScript library for making tile-based interactive map
2	Acapela	Acapela is a Voice as a Service provider. The service offers text to speech solutions to give voice to content in up to 25 languages and up to 50 voices. The Acapela API lets developers integrate speech into their application and control the voice generat
3	Cohuman	Cohuman is a task-centric, team productivity tool that helps users coordinate and plan their daily tasks to effectively complete projects on time. The Cohuman API allows anyone to develop applications for the web, mobile devices and the desktop. With the API users can make a new task, assign the task to a person or to a team of people, and add content to the task by starting a conversation thread, attaching a file, or scheduling a due date. The API uses RESTful protocol and responses are formatted in XML and JSON.

4.2 Determination of the Number of Clusters

Because the number of Mashup categories is unknown in our dataset, we firstly conduct an experiment to determine the number of clusters, which is necessary in the following experiments.

Table 3. The information of Mashups

MashupID	Name	APIs	Tags	Description
1	5th Bar Phone Reviews	Amazon Product Advertising, CNET, eBay, YouTube	auction, mobile, shopping, video	5th Bar is a new way to avoid getting the wrong phone again. Find and share reviews and information about mobile phones, cell phone carriers, and accessories. Mashup content from YouTube, eBay, Amazon, and CNET.
2	A World of Nirvana	Google Maps, YouTube	mapping, music, nirvana, video	Dynamic tribute to Kurt Cobain, showing Nirvana live concerts on a Google map by year. Search for videos directly by keywords.
3	a.placebetween.us	Google Maps	events, mapping, social, travel	a.placebetween.us aims to simplify the task of finding a place to meet your friends. Provide your addresses and the type of place you want to meet at, such as coffee, diner, or movie and a.placebetween.us does the rest.

Table 4. Overview of the dataset

The total number of Mashups	5955
The total number of APIs used by Mashups	1069
The average number of APIs invoked by per Mashup	2.017
The minimum number of APIs in a Mashup	1
The maximum number of APIs in a Mashup	17

We select 85 % Mashups as training data and the rest as testing data. We then cluster these Mashups into multiple Mashup clusters with a number from 10 to 100 at a step 10 and employ the manifold ranking algorithm to recommend APIs. Finally, we evaluate the performance of the recommendation list using F-measure. The result is shown in Fig. 2.

Figure 2 shows a trend of decrease after an initial increase. When the cluster number reaches 40, F-measure obtains the maximal value. Therefore, in the following experiments, we set the number of clusters to be 40.

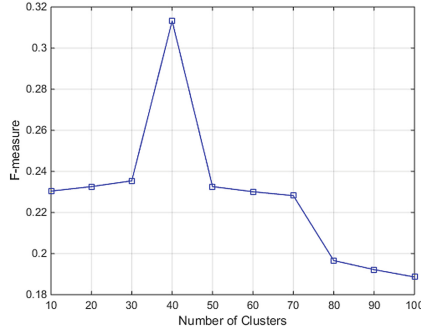


Fig. 2. Impact of number of Mashup clusters

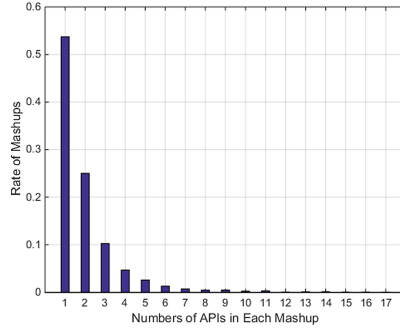


Fig. 3. Statistical analysis of the dataset

4.3 Evaluation Metrics

Before evaluating the quality of our approach, we make some statistical analysis on the dataset that we use. Figure 3 presents the relationship between the number of APIs in each Mashup and the rate of Mashups.

As we can see, over 88.9 % Mashups invoke few than 3 APIs. In the whole dataset, the average number of API used by Mashups is 2. It turns out that the usage of APIs is considerable low. Therefore, it is essential to recommend a list of APIs for Mashup developers.

Herein, we use the three measures of precision, recall, F-measure to evaluate the performance of our approach. The evaluation metrics are defined as follows.

- Precision:

$$p = \frac{|T_A \cap E_A|}{|T_A|} \quad (19)$$

where, T_A is the API recommendation result list, E_A is the testing Mashup actually used APIs, $|T_A|$ is the size of the API recommendation result list.

- Recall:

$$r = \frac{|T_A \cap E_A|}{|E_A|} \quad (20)$$

where, $|E_A|$ is the number of APIs the testing Mashup actually used.

- F-measure:

$$F = \frac{2 * p * r}{p + r} \quad (21)$$

It is a comprehensive evaluation on the precision and recall.

4.4 Performance of Our Approach

The API recommendation performance is likely to be influenced by data density. Data density means how many records in the matrix can be employed. In order to study the impact of training data density, in this experiment, we randomly remove 10 % to 90 % data from the original data as the training dataset, and use the rest data as testing dataset. There are a few parameters in our multi-relation based manifold ranking algorithm (i.e. *DMRrank*). These parameters control the weight of different terms in the manifold ranking formula. By adjusting these parameters, the approach can generate different recommendation lists. In our approach, we set $\rho = 0.1$, $\alpha = 0.1$; $\beta = 0.1$; $\theta = 0.1$; $\lambda = 0.2$; $\gamma = 0.3$; $\mu = 0.1$; $\eta = 0.1$. With these settings, our approach can perform better than other settings. We compare our approach with one of the state-of-the-art approaches, i.e. *GMrank* [13], which used manifold learning as well. We set the parameters of *GMrank* as $\alpha = 0.4$, $\beta = 0.1$, $\gamma = 0.1$, $\mu = 0.1$, $\eta = 0.3$, so that *GMrank* can get the best performance on our dataset. Moreover, in order to compare the impact on the number of selected Mashup clusters, we compare *DMRrank* with another approach (i.e. *MRrank*) which is just like *DMRrank* but simply selects the most similar Mashup cluster according to the user's requirement. Namely, *DMRrank* chooses one more similar Mashup cluster than *MRrank* and other settings are just keeping the same.

Figures 4, 5 and 6 present the precision, recall and F-measure comparisons on the training data with different density. As we can see, with the increasing density of training data, the precision, recall and F-measure values also grow. This observation means that larger density data is better for recommending APIs. Moreover, our approach outperforms the *GMrank* and *MRrank* approach in all cases. The results also show, *DMRrank* is better than *MRrank*, and *MRrank* is better than *GMrank*. It means that considering the composition and potential composition relations between APIs and selecting the two most similar Mashup clusters is indeed helpful for recommending APIs.

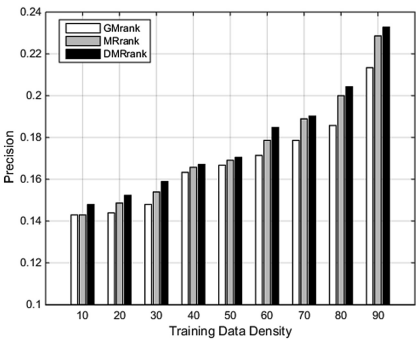


Fig. 4. Precision comparison

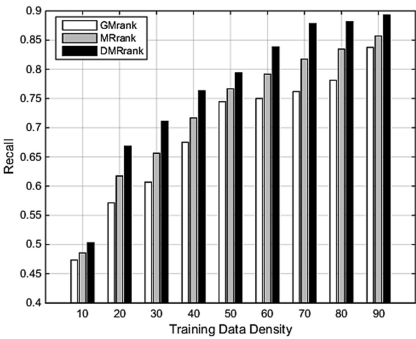


Fig. 5. Recall comparison

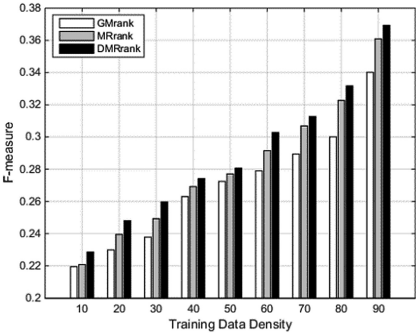


Fig. 6. F-measure comparison

5 Related Work

In this section, we survey related work on service ranking and recommendation.

There are several research work focusing on service ranking. Almulla et al. [14] presented a new Web services selection model based on fuzzy logic and proposed a new fuzzy ranking algorithm based on the dependencies between proposed qualities attributes. Jeh and Widom [15] designed a general similarity measure called SimRank, which is based on a simple and intuitive graph-theoretic model and defines the similarity between two vertices in a graph by their neighbourhood similarity. Mei et al. [16] proposed a ranking approach called DivRank, which is based on a reinforced random walk in an information network. It can automatically balance the prestige and the diversity of the top ranked vertices in a principled way. Tong et al. [17] proposed a goodness measure for a given top- K ranking list. It can capture both the relevance and the diversity for a given ranking list. Zhou et al. [18] proposed a unified neighborhood random walk distance measure called ServiceRank, which integrates various types of links and vertex attributes by a local optimal weight assignment to tightly integrate ranking and clustering by mutually and simultaneously enhancing each other.

With the increasing number of services in the Internet, service recommendation has become a hot topic in recent years. Li et al. [19] proposed a relational topic model to characterize the relationship among Mashups, APIs and their links to assist Mashup creators by recommending a list of APIs that may be used to compose a required Mashup given descriptions of the Mashup. Gao et al. [13] designed a manifold ranking framework for API recommendation. They recommend APIs for each Mashup cluster using manifold ranking algorithm which incorporates the relationships between Mashups, between APIs and between Mashups and APIs. Huang et al. [20] developed a novel approach for recommending developers in terms of navigation and completion of Mashup components with a large-scale components repository. They model the relationships between Mashup components by a generic layered-graph model. Huang et al. [21] presented a service recommendation method that suggests both services and their compositions, in a time-sensitive manner. Xu et al. [22] proposed a novel social-aware service recommendation approach, where multi-dimensional social relationships among potential users, topics, Mashups, and services are described by a coupled matrix model. Zheng et al. [23] presented a collaborative filtering approach for predicting QoS values of Web services and making Web service recommendation by taking advantages of past usage experiences of service users.

6 Conclusion

In this paper, we study the problem of recommending suitable APIs satisfying users' need for Mashup creation. We present a multi-relation based manifold ranking algorithm to assist Mashup developers by recommending a list of APIs that may be used to compose a required Mashup by giving a description of a Mashup. We firstly cluster existing Mashups into groups according to their textual descriptions. Then, we consider multiple relations between Mashup clusters and between APIs. Next, we associate Mashup clusters with APIs based on the popularity of APIs. Finally, we employ

manifold ranking algorithm to recommend APIs that the user may need for Mashup creation and perform a set of experiments to validate our approach on a realistic dataset. Experimental results validate the effectiveness of the proposed approach in terms of precision, recall, and F-measure and show that our approach outperformed the baseline approach for this particular dataset.

In future work, we would like to take the information of services providers and services users into consideration, and along with their potential relationships. Matrix factorization is a well-known service recommend method, so we will incorporate matrix factorization to our manifold ranking algorithm to get better recommendation performance.

Acknowledgments. The work described in this paper was supported by the National Natural Science Foundation of China under grant No. 61572186, 61572187, 61402168 and 61300129, Scientific Research Fund of Hunan Provincial Education Department of China under grant 15K043, 16K030, Hunan Provincial University Innovation Platform Open Fund Project of China under grant No. 14K037.

References

1. Fichter, D.: What Is a Mashup? <http://books.infotoday.com/books/Engard/Engard-Sample-Chapter.pdf>. Accessed 12 August 2013
2. Greenshpan, O., Milo, T., Polyzotis, N.: Autocompletion for Mashups. In: Proceedings of VLDB Endowment, Lyon, France, pp. 538–549 (2009)
3. Chen, L., Wu, J., Jian, H., et al.: Instant recommendation for web services composition. *IEEE Trans. Serv. Comput.* **7**(4), 586–598 (2014)
4. Huang, K., Fan, Y., Tan, W.: An empirical study of ProgrammableWeb: a network analysis on a Service-Mashup system. In: Proceedings of IEEE 19th International Conference on Web Services (ICWS), Honolulu, HI, pp. 552–559 (2012)
5. Kaufman, L., Rousseeuw, P.: Clustering by means of medoids. In: Dodge, Y. (ed.) *Statistical Data Analysis Based on the L1-Norm and Related Methods*, pp. 405–416. North-Holland (1987)
6. Singh, S.S., Chauhan, N.C.: K-means v/s K-medoids: a comparative study. In: Proceedings of National Conference on Recent Trends in Engineering & Technology, vol. 13 (2011)
7. Lü, L., Jin, C., Zhou, T.: Similarity index based on local paths for link prediction of complex networks. *Phys. Rev. E* **80**(4), 046122 (2009)
8. Zhou, T., Lü, L., Zhang, Y.: Predicting missing links via local information. *Eur. Phys. J. B Condens. Matter Complex Syst.* **71**(4), 623–630 (2009)
9. Breitenbach, M., Grudic, G.Z.: Clustering through ranking on manifolds. In: Proceedings of the 22nd International Conference on Machine Learning, pp. 73–80. ACM (2005)
10. Xu, B., Bu, J., Chen, C., et al.: Efficient manifold ranking for image retrieval. In: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 525–534. ACM (2011)
11. He, J., Li, M., Zhang, H.J., et al.: Manifold-ranking based image retrieval. In: Proceedings of the 12th Annual ACM International Conference on Multimedia, pp. 9–16. ACM (2004)
12. Zhou, D., Weston, J., Gretton, A., et al.: Ranking on data manifolds. In: *Advances in Neural Information Processing Systems*, vol. 16, pp. 169–176 (2004)

13. Gao, W., Chen, L., Wu, J., et al.: Manifold-learning based API recommendation for mashup creation. In: Proceedings of IEEE 22nd International Conference on Web Services (ICWS), pp. 432–439 (2015)
14. Almulla, M., Almatouri, K., Yahyaoui, H.: A qos-based fuzzy model for ranking real world web services. In: Proceedings of IEEE 21st International Conference on Web Services (ICWS), pp. 203–210 (2011)
15. Jeh, G., Widom, J.: SimRank: a measure of structural-context similarity. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 538–543. ACM (2002)
16. Mei, Q., Guo, J., Radev, D.: Divrank: the interplay of prestige and diversity in information networks. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1009–1018. ACM (2010)
17. Tong, H., He, J., Wen, Z., et al.: Diversified ranking on large graphs: an optimization viewpoint. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1028–1036. ACM (2011)
18. Zhou, Y., Liu, L., Perng, C.S., et al.: Ranking services by service network structure and service attributes. In: Proceedings of IEEE 20th International Conference on Web Services (ICWS), pp. 26–33 (2013)
19. Li, C., Zhang, R., Huai, J., et al.: A novel approach for API recommendation in Mashup development. In: Proceedings of IEEE 21st International Conference on Web Services (ICWS), pp. 289–296 (2014)
20. Huang, G., Ma, Y., Liu, X., et al.: Model-based automated navigation and composition of complex service Mashups. *IEEE Trans. Serv. Comput.* **8**(3), 494–506 (2015)
21. Huang, K., Fan, Y., Tan, W., et al.: Service recommendation in an evolving ecosystem: a link prediction approach. In: Proceedings of IEEE 20th International Conference on Web Services (ICWS), pp. 507–514 (2013)
22. Xu, W., Cao, J., Hu, L., et al.: A social-aware service recommendation approach for Mashup creation. In: Proceedings of IEEE 20th International Conference on Web Services (ICWS), pp. 107–114 (2013)
23. Zheng, Z., Ma, H., Lyu, M.R., King, I.: QoS-aware web service recommendation by collaborative filtering. *IEEE Trans. Serv. Comput.* **4**(2), 140–152 (2011)

Advances in Services Computing

10th Asia-Pacific Services Computing Conference,
APSCC 2016, Zhangjiajie, China, November 16-18,
2016, Proceedings

Wang, G.; Han, Y.; Martínez Pérez, G. (Eds.)

2016, XIII, 518 p. 230 illus., Softcover

ISBN: 978-3-319-49177-6