Mashup by Surfing a Web of Data APIs

Huajun Chen[1], Bin Lu[1], Yuan Ni[2], Guotong Xie[2], Chunying Zhou[1], Jinhua Mi[1], Zhaohui Wu[1]

[1] College of Computer Science,
Zhejiang University,
Hangzhou, 310027, China

{huajunsir, lb, cyzhou, mjh, wzh}@zju.edu.cn

[2] IBM China Research Lab,
Zhongguancun Software Park,
Beijing, 100193, China

{niyuan, xieguot}@cn.ibm.com

ABSTRACT

We present sMash, a system for facilitating users to mashup Web data. The aspects emphasized by the demo are: (1) how to help novice users master data APIs and relationships amongst them easily; (2) how to inspire various users to build more amazing Web data mashups. First, a real-life data API network is constructed and visualized to enable users to surf and mashup. Second, two kinds of recommendations are generated dynamically based on a comprehensive analysis of the network, user's traces and a repository of mashups to provide navigation.

1. INTRODUCTION

Proliferation available data APIs[1][1] give rise to phenomenal growth of mashup [2, 3] applications on the Web. However, what accompanies with this are the problems that users encounter during mashup building, including (1) the difficulties to locate and combine the right data APIs because of lack of knowledge about these APIs and relationships amongst them; (2) inconvenient and unpleasant steps to be performed whenever they want to mashup a new API, such as finding and reading its specification, deciphering the appropriate functions to be used, checking its mashupability with other APIs, etc; (3) limited ability to discover more mashupable APIs which may make their mashups more abundant.

sMash – short for semantic-based Mashup – is a novel Web data mashup system, which aims to resolve above three issues. By leveraging the power of semantic, social community and collective intelligence technologies, it links all the mashupable APIs together automatically to form a data API network and visualizes it smoothly to enable various users to build mashup easily and intuitively. Compared with current industrial mashup tools, including IBM Mashup Center [4], Yahoo Pipes [5], Microsoft Popfly [6] and Intel Mash Maker [7], it offers the following four unique features which cover several mashup scenarios that cannot be addressed well by them:

[1] We regard data APIs as any information source that can offer their data in a RESTful way.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the VLDB copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Very Large Database Endowment. To copy otherwise, or to republish, to post on servers or to redistribute to lists, requires a fee and/or special permissions from the publisher, ACM.
VLDB '09, August 24-28, 2009, Lyon, France.
Copyright 2009 VLDB Endowment, ACM 000-0-00000-000-0/00/00.

Feature 1: Automatic Generation of Mashup Graph. This can be applied to users who have clear ideas about which data APIs to use and how the mashup results should be like. By using current mashup tools, to construct a mashup graph, users need to use drag-and-drop tools to select APIs or specify URL of APIs; choose the appropriate functions; then connect them one by one. By using sMash, what users need to do is a little bit "fuzzy-match-keyword-search" for the name of APIs; then a mashup graph is generated automatically.

Feature 2: Surf and Mashup. This is convenient for users who have no clear purpose of their mashup results. All of the current mashup tools cannot address this scenario. In sMash, each API can be regarded as a Web page. By an interactive interface, users can surf the data API network just like surfing the Web and mashup their interested APIs by leaving a trace. sMash is clever enough to record all the traces to form a mashup graph and tell users this is the mashup they want to build.

Feature 3: Trace-based Recommendation. Users, especially novice end-users, may still feel confused during surfing since they have to face such a large size of APIs. sMash can quickly grasp users' intention by analyzing the traces they have kept up to now, and provide a real-time recommendation about top-10 most popular links to go next.

Feature 4: Inference-based Recommendation. This is a more interesting feature which may attract power users. It is motivated by two statistical results: (1) because of the difficulty of being discovered and mastered, about 4/5 data APIs are rarely used to build mashups even if they may supply more abundant information; (2) after mapping all the mashups of the repository[2] into a sample network constructed utilizing all the APIs used by these mashups, we find that only 30% links of the network are covered, as depicted in Figure 5(a) and Figure 5(b). Therefore, sMash exploits and visualizes the undiscovered APIs and links, and recommends top-10 links based on users' traces to encourage interested users to surf and mashup which may bring them amazing results.

In the next section, we provide a systematic overview by describing main components and demonstrate how the above four features are addressed in the design and implementation of sMash. A system evaluation is presented in Section 3. Section 4 describes the demonstration plan.

2. SMASH OVERVIEW

As depicted in Figure 1, the sMash system consists of two main components: user interface and sMash Server. Layered approach

[2] The repository is formed by extracting all the mashups from community of Yahoo Pipes and ProgrammableWeb.com

is adopted in the design of server, which ensures the flexibility and scalability of the system. In the subsequent sections, we briefly demonstrate how the network is visualized and interacts with users, how the network is constructed and how the links are recommended.

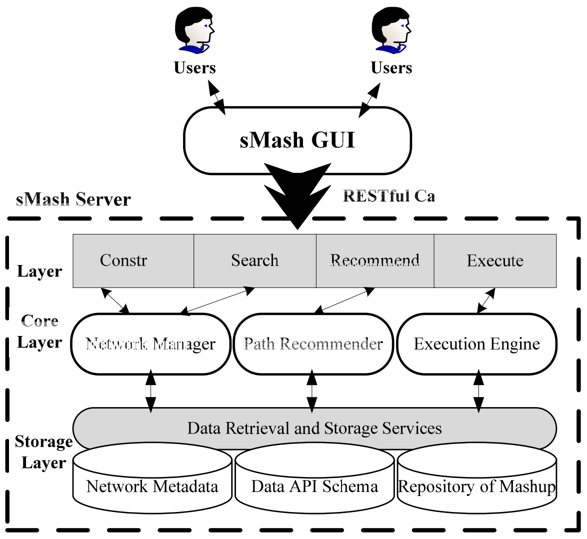


Figure 1: Architecture of sMash

2.1 Network Visualization and Interaction

Visualization of data API network is critical for better supporting the main features of sMash. Considering this key aspect, we developed a browser-based interface to enable users to experience the features easily and intuitively. The interface communicates with the server through a set of RESTful API interfaces.

How to enable automatic generation of mashup graph? A powerful query engine is provided for users to find the right APIs quickly. It can interact with users in real-time to suggest several candidate APIs by analyzing their entered content. Besides, a list of frequently used APIs is provided for fast selection. Then the selected APIs are visualized and mashupable ones are linked automatically to form a mashup graph. A little pruning needs to be done to remove redundant links if necessary.

How to enable surfing and mashup? At the beginning, an initial view of the visualized network is shown in the form of graph, in which each vertex represents one API and an edge between two APIs means they are mashupable. Several details have been paid attention to during implementation, such as (1) frequently used APIs are positioned closer to user specified API for fast locating; (2) a set of operations, such as zooming, moving and full screen, are developed to allow users to view this graph conveniently; (3) only the APIs mashupable with user specified API are retrieved and visualized in the initial view in case of making user feel overwhelmed and two operations, show/hide mashupable APIs are provided to facilitate users to extend/reduce this graph as they wish.

The smooth interaction enables users to navigate to an API easily to browse its information, such as an introduction about its usage, its functions and its linked (mashupable) APIs, an explicit explanation for why they are mashupable as well as the friendly navigation links to them, etc. To mashup their interested APIs, users just leave a trace by choosing the corresponding link. Besides, some operations are offered to facilitate users to manage their traces easily, such as the preview feature which can help users to see the data flow of their traces at any point in the process and the modification functionality which may enable users to update/erase their traces.

How to enable trace-based and inference-based recommendation? Along with the features mentioned above, two kinds of recommendations are provided for users. A recommendation item can either suggest changing the data flow of their current traces or adding some new APIs to extend the traces. It consists of four parts: (1) modified/added API names and data flow identifier used to express what is suggested; (2) a brief description to help users decide whether this suggestion is what they need; (3) a list of mashupable functions to describe the details; (4) a visualized source to show users how their traces will be like if they accept this item. A "clone" operation is offered for fast updating of their traces according to the recommendation. Besides, as an additional feature, how most of other users dealt with current traces are extracted from repository and visualized for users to refer.

A Typical Scenario: The screenshot in Figure 2 shows a typical scenario when a tourist wishing to build a mashup about beach is surfing the partial visualized network. After navigating to some APIs near "Flickr" and browsing the detailed information, she/he masters these APIs as well as the relationships amongst them quickly and comes up with a good mashup idea which is finding events, the latest posts about beach from "Upcoming", "Twitter" and "GoogleSearch", then taking the title of each result as input to search for related photos from "Flickr" and finally displaying these photos on "GoogleMap". With the help of friendly interactivity of the network, this idea is quickly realized. Next, by enabling the trace-based recommendation functionality, she/he receives a list of most related links which may bring her/him more inspiration. Taking "Youtube $\rightarrow$ GoogleMap" for example, it can make her/his final mashup result integrate not only the photos about beach but also the related videos. Gradually, by means of surfing and recommendation, her/his mashup becomes more and more abundant.

After completing graph building, a few parameters need to be set for each function of APIs. For convenience, sMash does an automatic mapping between matched parameters of functions. For example, the output values of latitude and longitude of "Flickr.getGeotaggedPhotos()" are automatically mapped into the corresponding input parameters of "GoogleMap.addPushpin()".

2.2 Network Construction

In data API network, an API is represented as a node; a link between two APIs means they have the mashupable relationship. Three main design principles have been considered carefully for its construction:

Precise Representation of Data API Metadata: As the main component of the network, each API needs a precise way to describe its metadata. Consequently we take advantage of the descriptiveness of RDF model to incorporate rich semantics of metadata (schema) of API, including tags, category and data content. Furthermore, microformats-like [8] frequently used semantic data types, such as "geo", "photo" and "event", are defined in order to describe data content precisely.

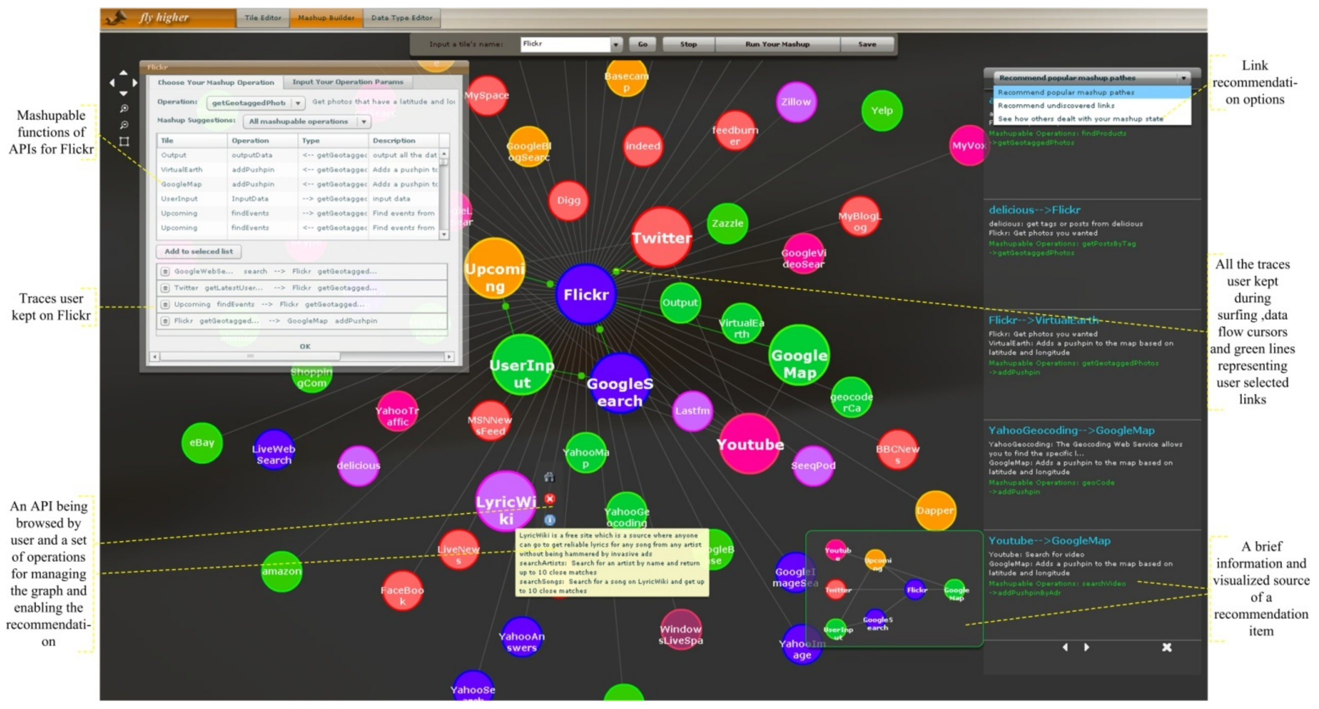

fly higher
Tile Editor
Mashup Builder
Data Type Editor
Input a tile's name:
Flickr
Go
Stop
Run Your Mashup
Save
Mashupable functions of APIs for Flickr
Traces user kept on Flickr
An API being browsed by user and a set of operations for managing the graph and enabling the recommendati-on
Link recommendati-on options
Recommend popular mashup pathes
Recommend undiscovered links
See how others dealt with your mashup state
All the traces user kept during surfing ,data flow cursors and green lines representing user selected links
A brief information and visualized source of a recommendation item
delicious-->Flickr
Flickr-->VirtualEarth
YahooGeocoding-->GoogleMap
Youtube-->GoogleMap
Flickr
Twitter
GoogleSearch
UserInput
GoogleMap
Youtube
LyricWiki

Figure 2: Screenshot of sMash Interface

Integrity of Links: Web data mashup is a creative activity and users may think of various novel ideas to combine a set of APIs. Therefore, in order to provide a better platform for users to exert their imagination, the network must be able to cover all the possible links. sMash considers that two APIs can be linked if the data contents of any function pairs between them match.

Scalability of Network: We make use of social aspect of web community and develop two auxiliary user friendly tools, API schema editor and data type editor, to enable users, especially developers, to contribute the schemas of data APIs as well as semantic data types to ensure the scalability of the network. At the time of writing, we have collected more than 350 APIs using these editors.

2.3 Link Recommendation

sMash's recommender builds upon previous work, utilizing a conditional probability calculation in conjunction with some efficient preprocessing.

Firstly, by means of a smart indexing of network, candidate links are fast retrieved by searching for the links containing any API in user's traces instead of filtering all the links. Secondly, suppose $\{B_1, B_2, ..., B_s\}$, s are involved links and size[3] of user's traces respectively, then the probability that a candidate link A can be added into user's traces equals to the probability that A is included by any mashup in repository given that any of the links are included, which is calculated by:

$$p_{final}(A)=\begin{cases}\dfrac{numOfMashupsContainingA}{sizeOfRepository}, s=0\\ P(A\,|\bigcup_{i=1}^{s}B_i)=\sum_{k=1}^{s}(-1)^{k-1}\sum_{I\subset\{1,...,s\}|I|=k}p(A\,|\,B_I), s>0\end{cases}$$

[3] The size refers to the number of links in user's traces.

Top-10 candidate links are recommended for trace-based recommendation. For inference-based recommendation, first, sMash collects the links with probability close to zero. Second, sMash infers that a link is meaningful if its features, such as tags and categories of its linked APIs, satisfy the features extracted from all the links in repository. Finally, sMash calculates the priority of each meaningful link based on the popularity of linked APIs and recommends top-10 to users.

To provide top-10 existing mashups about how others dealt with user's current traces sMash calculates the similarity between M_1 and each mashup M_2 in repository according to: $S(M_1, M_2) = numOfCommonLinks/size(M_1)$.

3. RECOMMENDATION EVALUATION

High-quality and high-performance has been addressed in the design and development of the recommender and empirically evaluated.

To test the quality of trace-based recommendation, first, we uniformly select three test sets from repository and use the remaining ones as the new repository. The size s of each mashup in three test sets is 1, 2 and 3 respectively[4]. Second, a subgraph with size (s-1) of each tested mashup graph is selected as input to simulate user's current traces and it is considered a *hit* if the last link in each graph is included by the recommended links. Taking the ranking into consideration, we measure the quality according to the following formula:

$$\Pr ecision=\frac{\sum_{i=1}^{numofHits}(1.1-ranking_i/10)}{numOfTests}, ranking_i \le 10.$$

[4] According to our statistical result, mashups with size less than 3 cover more than 85% of the repository, so the selected three test sets are sufficient for testing.

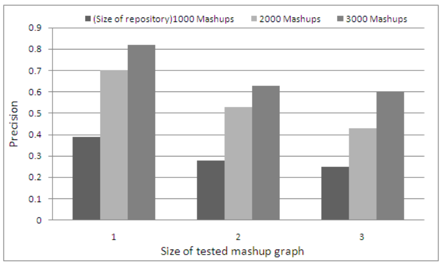


Figure 3: Quality of Trace-based Recommendation

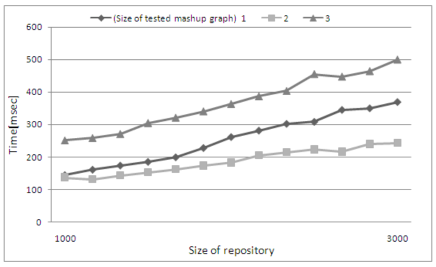
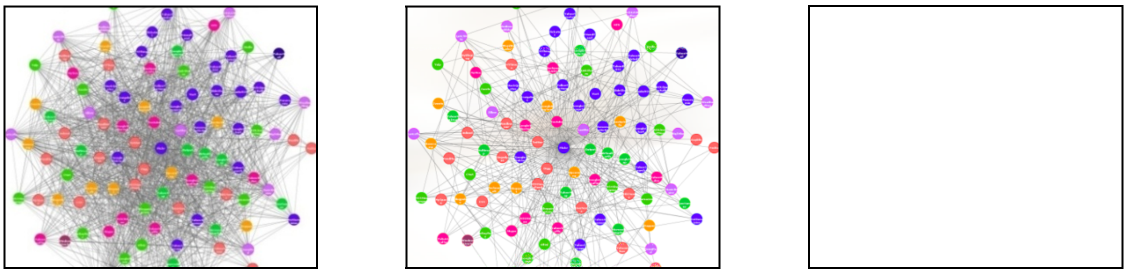


Figure 4: Recommender Performance



(a) **(b)** **(c)**

Figure 5: Quality Evaluation of Inference-based Recommendation. (a) shows the sample data API network composed of 112 APIs and 3814 links; (b) is composed of the 112 APIs and all the links that have been used by users to build mashups so far, only 1123 links in all; (c) is a combination of (b) and the recommended links which are obtained by taking part of each mashup in repository as input to simulate users' traces, 2288 links in all.

Figure 3 and 4 shows the quality and performance respectively across the test sets over the repositories with different size. As we can see, the recommender can keep high-quality with reasonable processing time and an appropriate repository size. Besides, since the recommendations are the most popular and relevant links users might not think of, they may build more amazing mashups with the help of them.

The evaluation of inference-based recommendation is to show that it can provide users with relevant but undiscovered links. The result is shown in Figure 5(c). As can be seen from the figure, given a set of APIs, about 50% new links[5] are recommended, making a very large space for users to play.

4. DEMONSTRATION

The Goal of Demonstration: Through the demo, we plan to show the following two aspects of sMash: (1) The visualized network can help users master data APIs and relationships amongst them easily and intuitively; (2) The network combined with link recommendations can inspire users to build more abundant Web data mashups that may go beyond their imagination.

The way of Demonstration: The GUI of sMash is Web-based which is accessible at http://www.dart.zju.edu.cn/mashup. In the demonstration session, we plan to let participants interact with this system directly, especially experience different mashup scenarios which are addressed by the features of sMash. Users who have a particular scenario in their mind can experience the automatic generation of mashup graph. For participants new to mashup, they can surf the network starting from their familiar information sources, such as Flickr, Facebook and Twitter, and build some simple mashups under the guidance of trace-based recommendation. Power users may be surprised at finding so many mashupable links around their frequently used APIs which they have never realized by means of inference-based recommendation.

We also plan to demonstrate the detailed procedure of the generation of Figure 5(b), thereby illustrating some interesting statistical results of current mashup state on the Web, for example, which links or combination of links among popular APIs are the most/least frequently used. Besides, we hope to discuss with other participants to see how the visualization and recommendation should be improved and what additional features are needed in sMash to further refine the design and implementation.

[5] We have manually looked over a random sample of few hundred links, and found that most of them make sense and are related with users' intention to a certain extent.

5. ACKNOWLEDGMENTS

This work is funded in part by No.NSFC60525202, China 863 project No.2006AA01A123, Zhejiang University and IBM CRL joint project called social enabled semantic data mashup.

6. REFERENCES

[1] ProgrammableWeb: http://www.programmableweb.com

[2] D. Butler. Mashups mix data into global service. Nature, 439(5):6–7, 2006.

[3] A. Jhingran. Enterprise information mashups: Integrating information, simply. In VLDB, pages 3–4, 2006.

[4] IBM Mashup Center: https://greenhouse.lotus.com/home/product.jsp?p=mashups

[5] Yahoo Pipes: http://pipes.yahoo.com

[6] Microsoft Popfly: http://www.popfly.com/

[7] Intel Mash Maker: http://mashmaker.intel.com/

[8] Microformats: http://microformats.org/