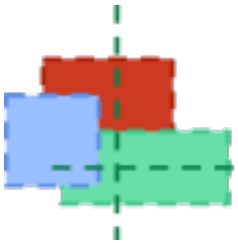
UNIVERSITY OF
ALBERTA

Web Service Matching for RESTful Web Services



R. Rabbany K., E. Stroulia, O. R. Zaiane
Computing Science Department
University of Alberta
Edmonton, Canada
{rabbanyk,stroulia,zaiane}@ualberta.ca

Presented by: ...

13th IEEE International Symposium on Web Systems Evolution (WSE 2011)

Introduction

Growing number of web services & wide range of services

- new challenges in the field of software engineering

service discovery, integration, composition & *service matching*

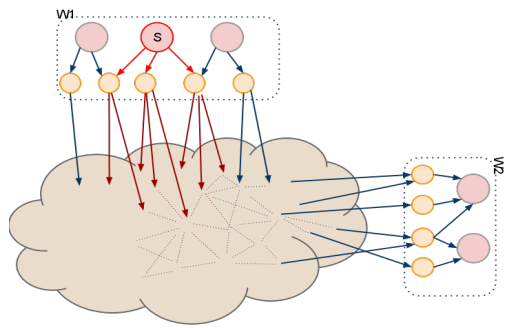
Web service matching (alignment)

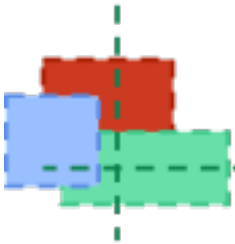
Mapping the functionalities of two web services

Semantic Flow Matching (SFM)

Graph-theoretic approach for matching REST web services

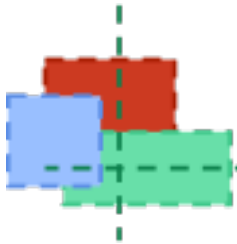
Heterogeneous network of WADL elements and semantically related terms & uses this network to match similar functionalities of web services





Outline

- Motivation
  - Why Web Service Matching?
  - Example Scenario
  - Why Rest, Why WADL?
- Background
- Method
- Experiments
- Conclusions



Motivation - Why Web Service Matching?

Redundancy in web-service ecosystem

Multitude of APIs provide a wide range of different services with a substantial semantic overlap i.e. *many of them provide the same functions*

Overlapping functionalities

enables developers to migrate from one API to anther when the API originally used becomes unavailable or insufficient for their needs

To support the discovery of similar APIs, several methods are being developed for web-service discovery

- Query-based methods relying on keywords and identifiers

Dong et. al. VLDB ’04, Wang et. al. WISE ’03

- Clustering

Nayak et. al. WI ’07

- Structure matching

Motahari Nezhad et. al. WWW ’10, Mikhaiel et. al. ICSOC ’07

Motivation - Example Scenario

Service migration

Clients that are using different web services for the same task, should easily be able to share their data, migrate from one service to the other in the case of service deprecation and also have a single point of access to their data distributed over different web services.

Del.icio.us for sale by Yahoo!

del.icio.us

many users worried about their data

several other bookmarking sites provide import

Automatically matching Del.icio.us and other similar service

Simplify the migration

Also Enable users to use Del.icio.us with Diigo, Zootool and Hbookmark, etc. at the same time and with a single point interface

Zootool hbookmark diigo PINBOARD

Motivation - Why Rest, Why WADL?

Service-matching techniques

- Interface matching (our focus)
  - Common interface-description language e.g. WSDL (Web Service Description Language)
- Behaviour matching
  - Usage context around the services, as specified in client applications or BPEL (Business Process Execution Language) descriptions
- Combination of both (which is more likely to produce better result)

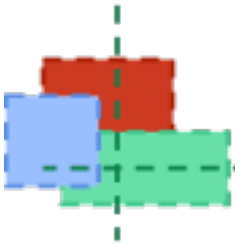
Increasing trend of Web 2.0 applications from SOAP-based web services to RESTful web services

WADL (Web Application Description Language)

Description for RESTful web services i.e. alternative for WSDL descrip
SOAP services

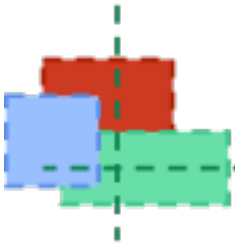
Our Contribution

Focusing on interface matching, we propose a novel method to match functionalities of two given RESTful web services based on their WADL description/interface by building a network to capture their semantic relation.



Outline

- Motivation
- Background
  - Related Works
  - Rest & WADL
- Method
- Experiments
- Conclusions



Related Work

Similar matching problems

- Schema Matching in Databases

Matches schemes based on their predicted semantics

  - Different granularity: components of two given schema
  - Highly matched Web services are loosely related
  - Different kind of Information

- Text Document Matching in Information Retrieval

Term-doc frequencies

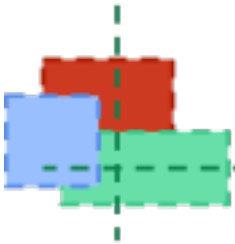
  - Web service docs are highly compact
  - Ignore much information, e.g. structure

- Software Component Matching

Signature mapping

  - Different level of expressiveness

Also related to Service Discovery



REST & WADL

REST: REpresentational State Transfer

Software architecture style

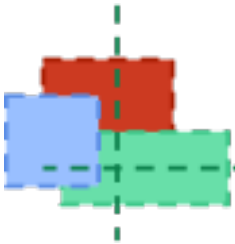
- Clients send requests, servers process & return responses
- Req & res: transfer of representations of resources
- Client: rests when interacting with user, goes to transition by sending request, transfers to new state by getting the response
- Description
  - WSDL to describe SOAP over HTTP
  - Abstraction purely on top of SOAP (e.g., WS-Transfer)
  - Without using SOAP at all.
  - **WADL: REST alternative to WSDL**

WADL: Web Application Description Language

Machine processable XML description for REST

- **set of resource elements**
- **param: describes the input**
- **method: describes the request and response of a resource**
  - **request: rep. of input: types, HTTP headers, etc.**
  - **response: rep.of service's response: JASON, XML, etc.**

Not yet widely in use



Outline

- Motivation
- Background
- Method
  - The Semantic Flow Matching
    - Overall Process
  - The Converter Module
    - An Example of the Generated WADL
  - The Mapper Module
- Experiments
- Conclusions

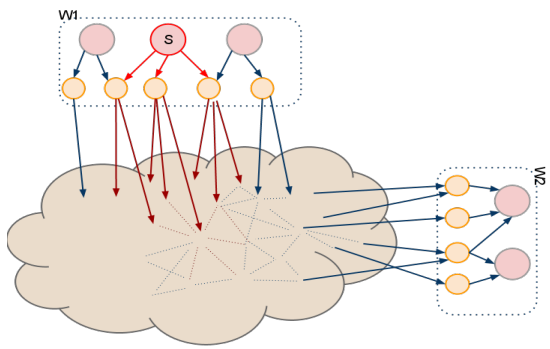
The Semantic Flow Matching

Semantic Flow Matching (SFM)

Graph-theoretic approach for matching REST web services

Incorporates linguistic knowledge and domain specific heuristics to automatically match given WADLs

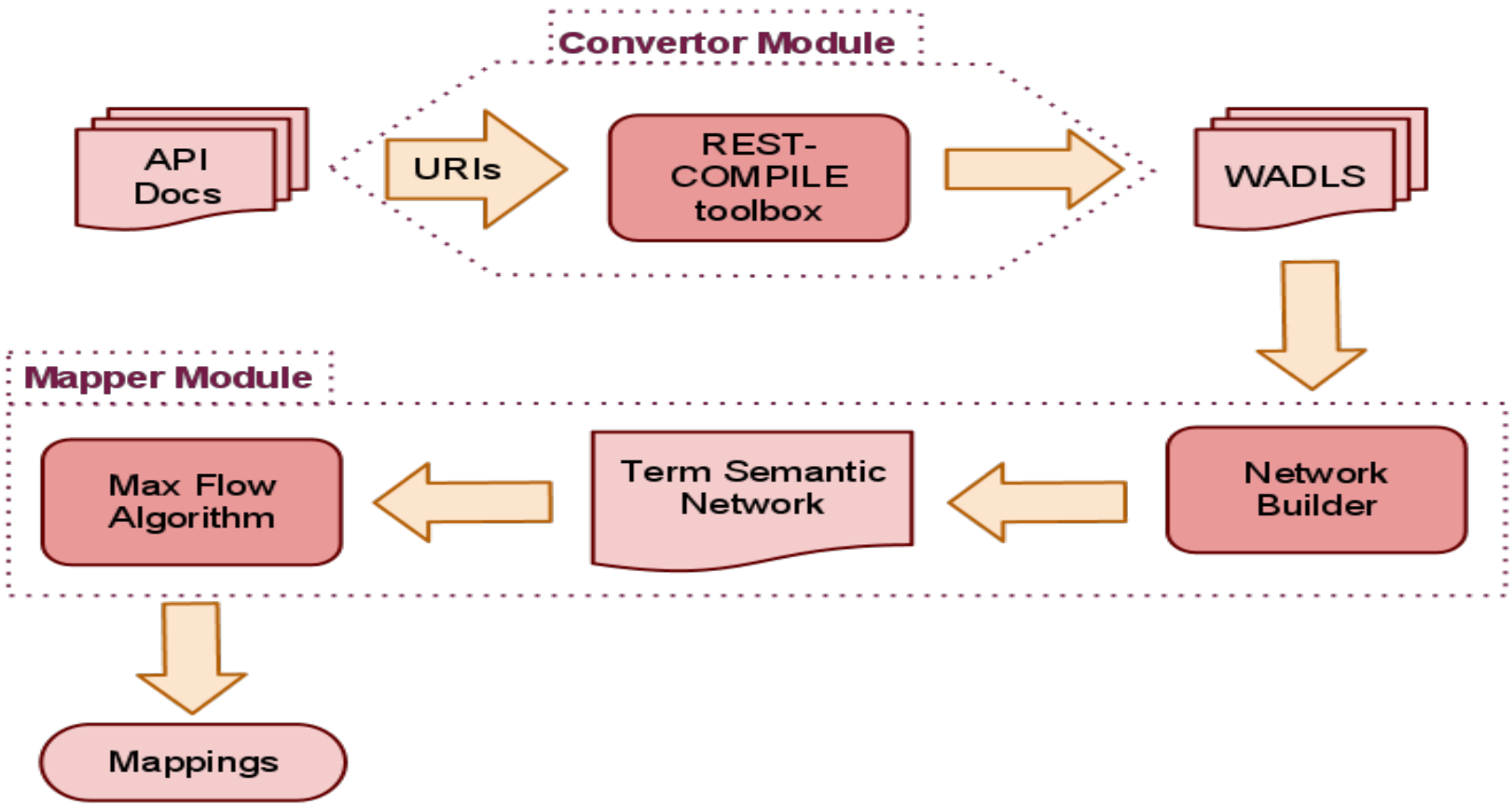
1. Creates a heterogeneous network of WADL elements and semantically related terms
2. Searches for the most similar methods (of different web services) by looking for the ones with maximum semantic flow between them



Overall Process

Consists of two modules:

- **Converter**: wraps the REST web services in WADL format
- **Mapper**: matches web services based on their semantics extracted from the WADL interface



The Converter Module

Transforms the HTML documentations of REST APIs into WADLs

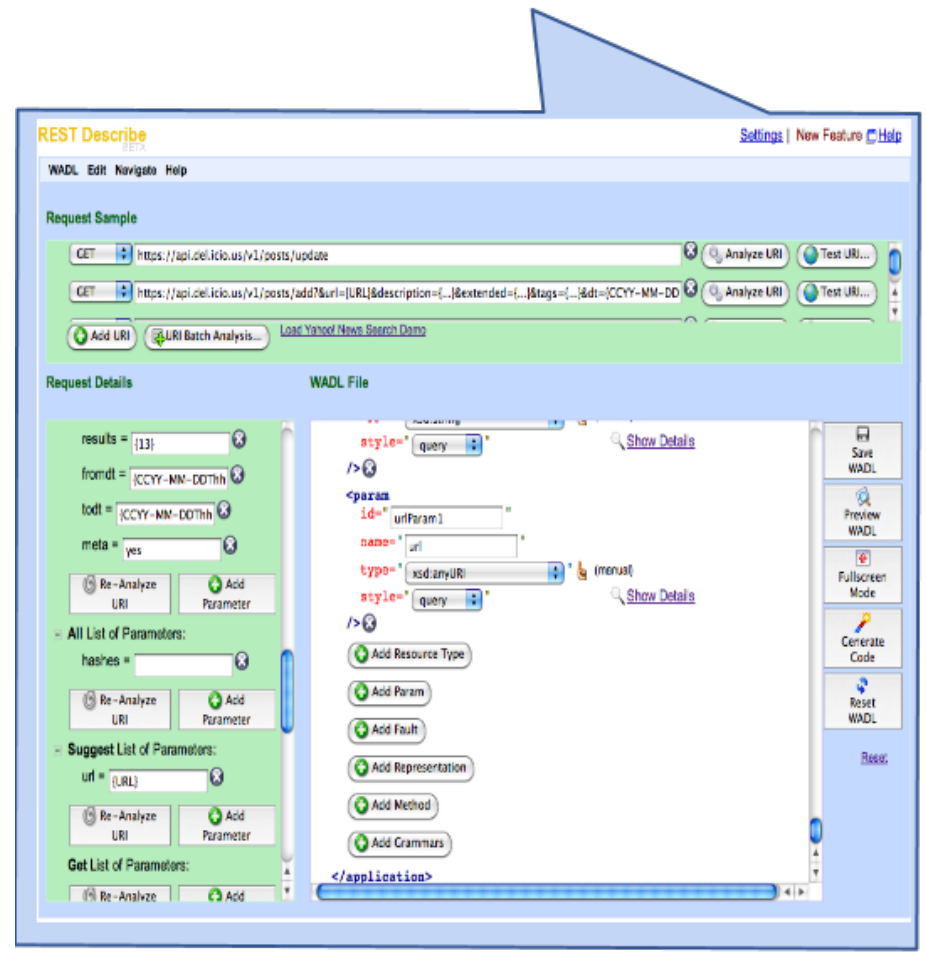
Google REST Describe:

Semi-automatic tool

Creates WADLs of web apps from sample request URIs

Needs manual modification

e.g. adding the responses & representations that explain the response media type

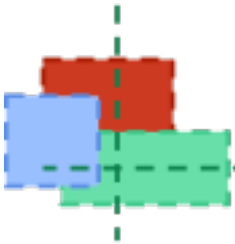


An Example of the Generated WADL

http://zootool.com/api/add/?url=http://www.google.com&title=Google&tags=search, google&description=searchEngine&public=y

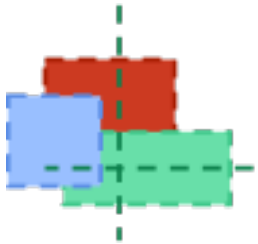
```
1   <?xml version="1.0" encoding="utf-8"?>
2   <application xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/
3       <resources base="http://zootool.com/">
4           <resource path="api">
5               <resource path="users">
6                   <resource path="items"> [17 lines]
24                  <resource path="info"> [11 lines]
36                  <resource path="validate"> [11 lines]
48                  <resource path="friends"> [14 lines]
63                  <resource path="followers"> [13 lines]
77                  <resource path="profiles"> [10 lines]
88              </resource>
89              <resource path="items"> [30 lines]
120             <resource path="add"> [22 lines]
143         </resource>
144     </resources>
145     <param name="username" type="xsd:string" style="quer
147     <param name="apikey" type="xsd:string" style="query"
149     <param name="offset" type="xsd:integer" style="query
151     <param name="limit" type="xsd:integer" style="query"
153     <param name="search" type="xsd:boolean" style="query
155 </application>
```

```
<resources base="http://zootool.com/">
    <resource path="api">
        <resource path="users"> [83 lines]
        <resource path="items"> [30 lines]
        <resource path="add">
            <method name="GET">
                <request>
                    <param name="url" type="xsd:anyURI"
                    <param name="title" type="xsd:string
                    <param href="#apikeyParam"/>
                    <param name="tags" type="xsd:string"
                    <param name="description" type="xsd:
                    <param name="referer" type="xsd:anyU
                    <param name="public" type="yesNo" st
                </request>
                <response> [2 lines]
            </method>
        </resource>
</resource>
```



Outline

- Motivation
- Background
- Method
  - The Converter Module
  - The Mapper Module
    - Problem
    - Solutions
    - SFM Approach
- Experiments
- Conclusions



The Mapper Module - Problem

Find a partial matching between groups of structured data
shallow tree of three levels -- resources, methods and parameters

The actual objective is to match the methods

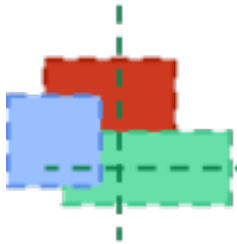
The other levels of structure in WADL Convey NO information about the functionalities of the API and are mostly for flexible API documentation and preventing repeated descriptions

Size of each web-service tree is small

Every web service provided a small set of functionalities

The matching is uncertain

Two methods may not have the exact same number of parameters
Parameter names do not match, not even synonyms
e.g. tag and bookmark



The Mapper Module - Solutions

Off-the-shelf Machine Learning

- Lack of labeled data i.e. already matched web-services
  - most informative features should be extracted manually
- Linguistic relations (similar names of two parameters) in feature design
  - Automated design of such feature set using statistical learning techniques is an standard approach in NLP, but manual design is deemed very difficult

Unsupervised techniques i.e. clustering

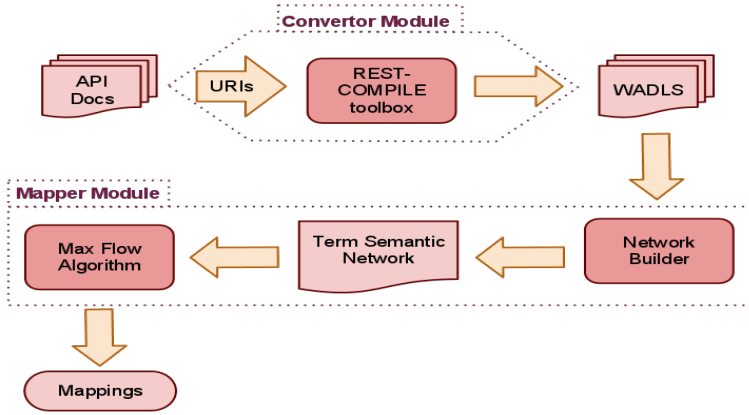
- Single member clusters in solution => undesirable in clustering
  - Methods in one service with no corresponding methods in the other service
- Methods of same service cluster together => violates the problem
  - As they share many similarities, such as common naming scheme

Semantic Flow Matching (SFM)

- Heuristic graph-theoretic matching approach
- Incorporates linguistic knowledge to find the best matching
- Works without labeled data (unsupervised)
- Leverages available toolsets in optimization (i.e. max-flow)

Outline

- Motivation
- Background
- Method
  - The WADL Converter Module
  - The Mapper Module
    - Problem
    - Solutions
    - SFM Approach
      - Overview
      - The SFM Network
        - Structure
        - Extraction
        - Example
      - Flow Based Matching
- Experiments
- Conclusions



SFM Approach - Overview

Matching two web services w_1 and w_2

SFM Network

Heterogeneous network of WADL elements and semantically related terms

By connecting

Every method element to its corresponding WADL elements

Each WADL element to their corresponding terms in the semantic network of synonym terms

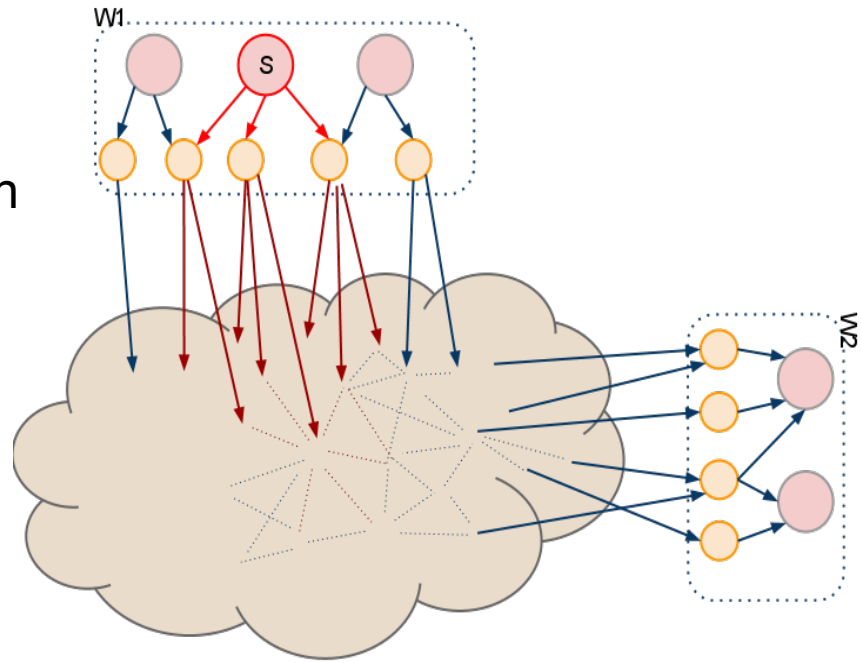
Flow Based Matching

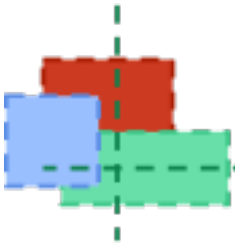
Find the proper match for s of w_1 in w_2

By Directing

Edges from/to w_1/w_2 outward/inward

Set s as a sink & find the method in w_2 that receives the maximum semantic flow





The SFM Network - Structure

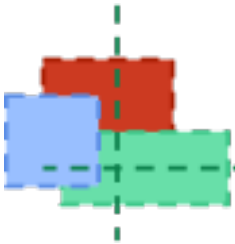
SFM network has two types of nodes

- WADL element nodes
  - Method, parameter and resource elements
  - Could be easily extended to include other elements as well
    - Optional doc: document functionality of methods in narrative text
    - Responses: include the media-type of the output

Each of the WADL element nodes are connected to one or more term nodes

- Term nodes
  - English terms in the synonymity cloud
  - Term nodes are connected to each other if they are synonyms

E.g. parameter element node with name attribute value `tags' is connected to `tag' term node in the synonymity cloud where it is connected to its synonym terms such as `label' and `mark'



The SFM Network - Extraction

Incorporated two kinds of knowledge

- Common natural-language processing knowledge

for node and edge discovery

- Porter stemmer in order to extract related terms to each WADL element
- WordNet for adding edges between synonym terms

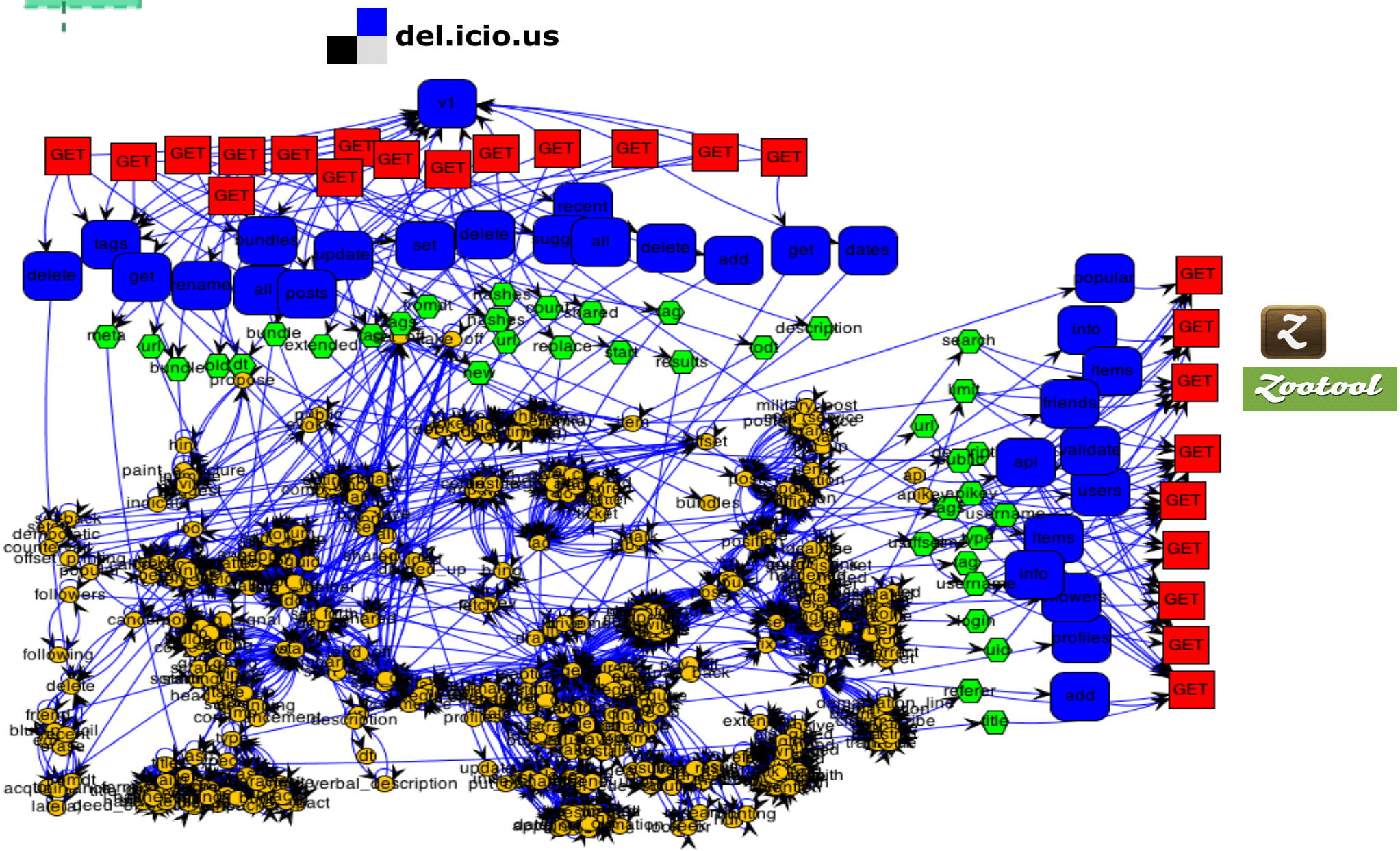
- Domain knowledge

for edge weighting following general heuristic rules

- *Edges from element nodes are weighted more than edges between term nodes*
- *Edges from method elements to resource elements are weighted more than edges to parameter elements*
- *Edges to required parameters are weighted more than edges to regular parameters*
- *Edges to resources of a method are weighted based on their path depth*

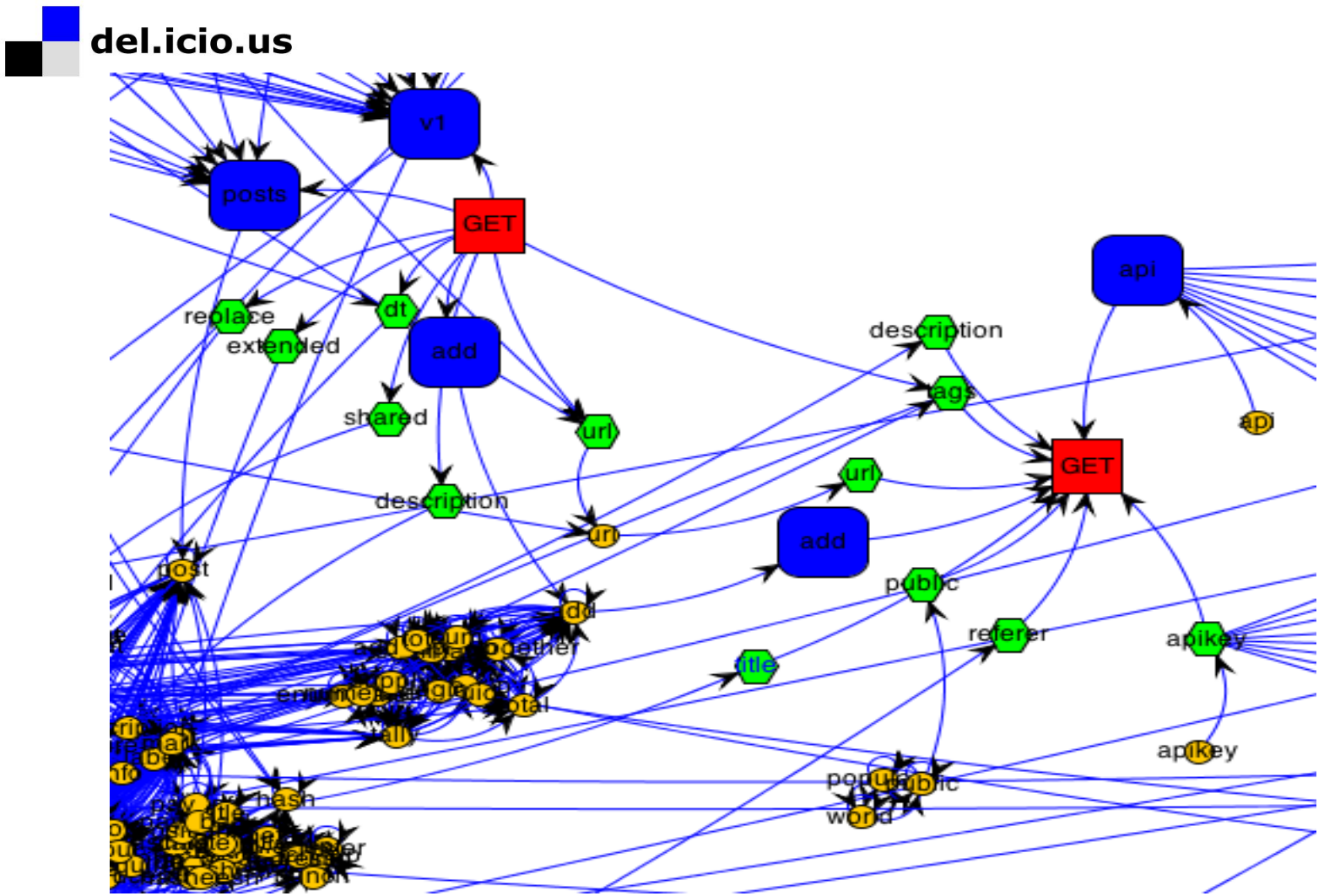
e.g. /api/tags/add => w(add) > w(tags)$

The SFM Network - Example



The SFM Network Example - Zoom

`add tags' method in Del.icio.us and `add' method in Zootool

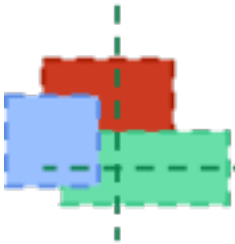


Flow Based Matching

Having the SFM network we use the flow based matching to find the possible matches between the methods of given services

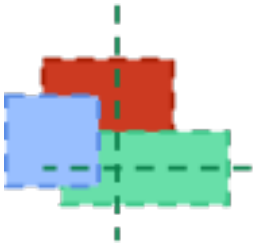
- Calculates maximum semantic flow from each m_1 in $wadl_1$ to every m_2 in $wadl_2$
  - Long-standing problem in optimization theory
  - Calculates the maximum flow from a source to a sink through the given flow network where edges have weight/capacity
  - Edmonds-Karp implementation based on the Ford-Fulkerson method, $O(|V| \cdot |E|^2)$
- Consider strong $<m_1, m_2>$ as a match
  - Stored sorted in M (set of possible matches)

{Matching the method elements}
for all method element m_1 in $wadl_1$ **do**
  for all method element m_2 in $wadl_2$ **do**
    $mf \leftarrow$ MAX-FLOW form m_1 to m_2
    if $mf > \delta$ **then**
      $ind \leftarrow 0$
      while $M(ind)_{mf} > mf$ **do**
        $ind \leftarrow ind + 1$
      end while
      insert $<< m_1, m_2 >, mf>$ in $M(ind)$
    end if
  end for
end for



Outline

- Motivation
- Background
- Method
- Experiments
  - Case Study Experiment
  - Case Study Results
- Conclusions



Case Study Experiment

del.icio.us

diigo

Zootool

hbookmark

Social bookmarking APIs

Del.icio.us, Diigo, Hbookmark and Zootool

*Obtain a **complete matching** for the all 6 pairs of these APIs*

Having M (possible matches of pairs of methods)

Each associated with a confidence value (actual flow between that pair)

Greedily choose the non-overlapping matches from M

starting with the match with the highest confidence value, and proceeding down the confidence values, keeping those that do not overlap (common methods) with any of the previous matches

Case Study Results

Diigo v.s. Zootool

SFM correctly maps the only two methods in the Diigo to their corresponding methods in Zootool

- `POST bookmark' method from Diigo with the `GET add' method from Zootool with a high confidence
- `GET bookmarks' method from Diigo with the `GET item' method from Zootool

https://secure.diigo.com/api/v2/bookmarks?user=joel&count=10&start=0&sort=1&tags=Spot,Film&filter=public&list=goodee https://secure.diigo.com/api/v2/bookmarks?url=www.diigo.com&title=Diigo+-+Web+Highlighter+and+Sticky+Notes&tags=diigo,bookmark,highlight&shared=yes&desc=sample&readLater=no
http://zootool.com/api/users/items/?username=bastian&apikey=123&login=true&tag=po&offset=2&limit=3 http://zootool.com/api/users/info/?username=bastian&apikey=123 http://zootool.com/api/users/validate/?username=bastian&apikey=123 http://zootool.com/api/users/friends/?username=bastian&apikey=123&offset=2&limit=3&search=true http://zootool.com/api/users/followers/?username=bastian&apikey=123&offset=2&limit=3&search=true http://zootool.com/api/users/profiles/?username=bastian&apikey=123 http://zootool.com/api/items/info/?uid=iw6og3&apikey=123 http://zootool.com/api/items/popular/?type=week&apikey=123 http://zootool.com/api/add/?url=http://www.google.com&title=Google&apikey=123&tags=search,google&description=searchEngine&referer=http://www.bing.com&public=y

possible match with MAX-FLOW of: **174** **POST bookmarks**: [title, shared, bookmarks, desc, v2, readLater, url, tags, api] & **GET add**:[public, apikey, description, title, api, referer, add, url, tags]
possible match with MAX-FLOW of: **76** **GET bookmarks**: [user, sort, bookmarks, v2, count, list, filter, tags, api, start] & **GET items**:[apikey, tag, username, limit, api, offset, users, items, login]

Case Study Results

2) Del.icio.us and Zootool

possible match with MAX-FLOW of: 170.0
GET add: [shared, url, extended, replace, posts, v1, description, dt, tags, add]
&
GET add:[public, apikey, description, title, add, referer, url, tags, api]

3) Del.icio.us and Hbookmark

possible match with MAX-FLOW of: 100.0
GET set: [bundles, tags, bundle, v1, tags, set]
&
GET tags:[page, user, rpp, tags, api, cb]

4) Del.icio.us and Diigo

possible match with MAX-FLOW of: 70.0
GET add: [shared, v1, url, extended, replace, posts, description, dt, tags, add]
&
POST bookmarks:[bookmarks, title, shared, desc, readLater, v2, api, url, tags]

possible match with MAX-FLOW of: 28.0
GET recent: [v1, count, posts, recent, tag]
&
GET bookmarks:[user, bookmarks, sort, count, v2, list, api, filter, tags, start]

5) Diigo and Hbookmark

possible match with MAX-FLOW of: 214.0
GET bookmarks: [user, v2, sort, bookmarks, api, count, list, filter, tags, start]
&
GET bookmarks:[page, user, rpp, bookmarks, cb, api]

Other five combinations of APIs

Very loosely coupled

complements instead of substitutions
e.g. Zootool is more of the users/followers v.s. Del.icio.us more for managing tags

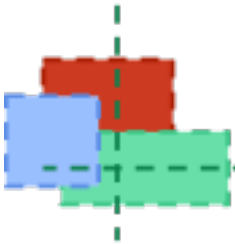
Very few possible matches with high confidence

6) Zootool and Hbookmark

possible match with MAX-FLOW of: 90.0
GET add: [public, apikey, description, title, referer, api, url, tags, add]
&
GET tagged:[tag, page, user, rpp, tagged, api, cb]

possible match with MAX-FLOW of: 74.0
GET followers: [apikey, search, limit, followers, username, offset, api, users]
&
GET search:[q, page, user, rpp, search, api, cb]

possible match with MAX-FLOW of: 74.0
GET items: [apikey, tag, username, limit, offset, items, api, users, login]
&
GET tags:[page, user, tags, rpp, api, cb]



Case Study Results

Big gap between confidence of highly matched methods and poorly matched methods

Threshold parameter could be set automatically (mean of all confidences)

Evaluation result for the SFM method

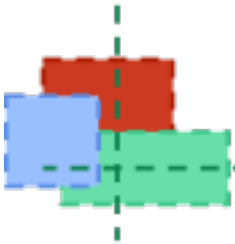
Precision:

# of correctly matched methods / total # of matched methods

Recall:

of correctly matched methods / total # of pairs that should be matched

APIs	precision	recall
Diigo and Zootool	1	1
Del.icio.us and Zootool	1	.33
Del.icio.us and Hbookmark	1	.5
Del.icio.us and Diigo	1	1
Diigo and Hbookmark	1	1
Zootool and Hbookmark	.33	1
TOTAL	.88±.25	.80±.28



{Conclus/Discuss/Quest}ions

The SFM approach

- *Novel graph-theoretic approach for matching similar methods of two given RESTful APIs based on their WADL descriptions*
- A solution to a more general problem, web-service matching
  - Earlier solutions to similar matching problems could not naively be applied here
- Designed specifically for REST API alignment
  - The SFM idea could be easily adopted to other semantic matching problems

Possible Future Works

- Include other sources of information
- Incorporate the behavioral knowledge (Business Process)
- Include domain-specific semantic knowledge
  - general purpose lexical references like WordNet are not suitable for our problem
    - e.g. `bookmark' and `tag' are synonym, however they are not synonymous in the general English sense
- More extensive evaluation for the SFM approach
  - Small data set <= difficulty of generating WADL files
  - Loosely coupled APIs => experiments on exact matching impossible

Questions?