Exploiting Social Tagging in Web API Search

Devis Bianchini, Valeria De Antonellis, and Michele Melchiori

Dept. of Information Engineering University of Brescia
Via Branze, 38 - 25123 Brescia Italy
{bianchin,deantone,melchior}@ing.unibs.it

Abstract. Given the huge number of available Web APIs, a web designer might take advantage of the "wisdom" or collective knowledge of the other developers who used the Web APIs for their own mashups. This knowledge may implicitly derive from the use of the Web APIs in similar mashups, or may be obtained through explicit social tagging and rating of Web APIs. In this paper, we propose to exploit this knowledge to implement advanced Web API search patterns, depending on the development scenario the web designer is acting in, namely the creation of a new mashup, the completion of an existing one or the substitution of one or more Web APIs within it.

1 Introduction

The exploitation of the "wisdom" or collective knowledge of the other developers to select the best Web APIs for developing a mashup is getting more and more relevant in the Web 2.0 [1,2]. Besides advanced Web API characterizations performed by combining categories and semantic tags [3] or technical features (e.g., protocols, data formats) [4], some researchers proposed to use the information about past mashups to let the designer of a new mashup "learn by examples" [2]. These approaches have the advantage of leveraging information from the Web API repositories. Nevertheless, the additional information brought by the so-called *social tagging* is something more: there is an implicit collective knowledge deriving from the co-occurrence of Web APIs in the same or similar mashups and an *explicit* social tagging and rating of Web APIs by developers who used them in the past.

In this paper, we exploit a new model for Web API search and ranking. The model includes a Web API characterization at different levels of complexity, namely (i) the Web API description in terms of categories and technical features, as extracted from public repositories, (ii) the aggregation of Web APIs into mashups performed by other developers (*implicit collective knowledge*), (iii) the semantic tagging and rating of Web APIs by developers who used them in their own mashups (*explicit collective knowledge*). In this paper, we are compliant with the `ProgrammableWeb` repository (`http://www.programmableweb.com/`), which offers a large, updated and highly populated catalogue of Web APIs to be selected and aggregated into mashups. Nevertheless, as we have underlined in [5], the search facilities of this repository reflect the ones of the (few) other

R. Meersman et al. (Eds.): OTM 2013, LNCS 8185, pp. 764–771, 2013.

available public registries. Semantic tagging is performed by relying on the system proposed in [3]. Specific similarity and ranking measures based on the model are described. Such metrics implement different Web API aggregation scenarios, namely the development of a new mashup, the completion of an existing mashup or the substitution of a Web API in a given mashup. With respect to the multi-perspective Web API model we proposed in [6], we stress here the distinction between the implicit and the explicit collective knowledge and we refine the measures for Web API search and ranking according to this distinction.

The strong point of our approach relies on the tuning of Web API matching and ranking according to different Web API features. In this sense, our approach is the first one that, taking into account so many aspects in Web API descriptions, combines them in a systematic way. Other approaches focus on a subset of such aspects, namely technical features [4] or collective knowledge coming from Web API use in existing web mashups [7], but they do not rely on other aspects such as ratings and designers' expertise. Exploitation of designers' ratings avoids Web API recommendation based only on the number of past mashups where it has been used, despite the inner quality of the Web API [2]. Moreover, our approach distinguishes among different Web API search targets (namely the creation of a new mashup, the completion of an existing one or the substitution of one or more Web APIs within it), performing a step forward with respect to approaches such as [8], which starts from a Web API model that is close to ours, but it is not designed to be differently tuned according to changing search targets.

The paper is organized as follows. Section 2 contains the definition of the Web API model we rely on. Metrics for Web API search and ranking are described in Section 3. Finally, Section 4 closes the paper.

2 Web API Model

The model we assume in this paper for Web API characterization is based on a representation that is commonly used for folksonomies [9], where user-based assignments of tags to resources are described. In this paper, users are Web designers, who assign both semantic tags and ratings to Web APIs as used in a given mashup. We formally define a *Web API repository* as follows.

Definition 1. *A Web API repository is a tuple* $\Re := \langle \Omega, \mathcal{M}, \mathcal{D}, \mathcal{S}, \Gamma, \Lambda, \mathcal{T}, \mu \rangle$*, where:*

- Ω*,* $\mathcal{M}$*,* $\mathcal{D}$*,* $\mathcal{S}$ *are finite sets, whose elements are the Web APIs, the mashups, the web designers and the semantic tags, respectively;*
- $\Gamma \subseteq \mathcal{M} \times \mathcal{D}$ *represents the set of ownerships of mashups by designers; specifically, if* $\langle m, d \rangle \in \Gamma$*, then the mashup* $m \in \mathcal{M}$ *is owned by the designer* $d \in \mathcal{D}$*;*
- $\Lambda \subseteq \Omega \times \mathcal{M}$ *is the set of pairs modeling the* `composedOf` *relationship between mashups and Web APIs; specifically, if* $\langle \mathcal{W}, m \rangle \in \Lambda$*, then the Web API* $\mathcal{W} \in \Omega$ *is included in the mashup* $m \in \mathcal{M}$*;*

- $\mathcal{T}$ *is a quaternary relation between the finite sets, that is,* $\mathcal{T} \subseteq \Omega \times \mathcal{M} \times \mathcal{D} \times \mathcal{S}$*, representing the assignment by designers of semantic tags to Web APIs with respect to a given mashup;*
- μ *is a function which represents a quantitative rating assigned by a designer* $d \in \mathcal{D}$ *to quantify how much the use of a Web API in* Ω *is suitable with respect to a given mashup* $m \in \mathcal{M}$ *which includes the Web API; the function is defined as* $\mu : \Omega \times \mathcal{M} \times \mathcal{D} \mapsto [0, 1]$*.*

The model enables a designer to assign semantic tags and ratings also with respect to mashups owned by other designers; however, ownership of mashups by designers might be relevant. For instance, ratings assigned by the mashup owner should be considered as more reliable. Ownership of mashups is not currently exploited in our Web API search and rating metrics and will be investigated as future work.

In our model, a Web API $\mathcal{W} \in \Omega$ is described by: (i) a name $n_{\mathcal{W}}$; (ii) a unique $URI_{\mathcal{W}}$; (iii) a set of categories $C_{\mathcal{W}}$; (iv) a set $\{t_{\mathcal{W}}\}$ of semantic tags; (v) a set $\mathcal{F}_{\mathcal{W}}$ of technical features, where a feature is a pair $\langle$`type`,{`values`}$\rangle$ (e.g., $\langle$`protocol`,{`SOAP`,`REST`}$\rangle$).

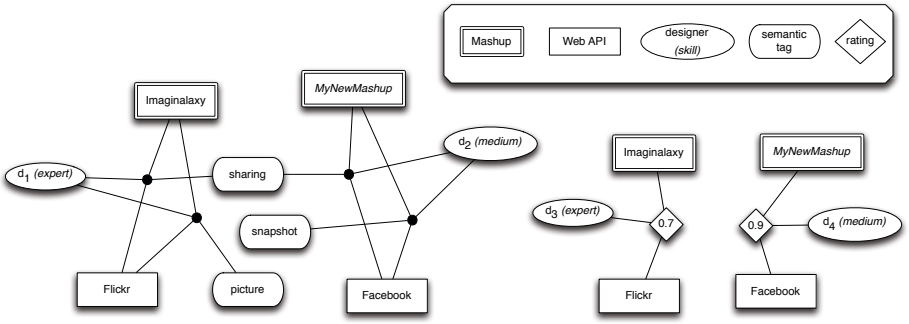
Each mashup $m \in \mathcal{M}$ is modeled through a name n_m and a Uniform Resource Identifier URI_m. No technical features are provided for mashups, since in public repositories these features are directly related to the component Web APIs.

Each designer $d_i \in \mathcal{D}$ is modeled through the skill $\sigma_i \in [0, 1]$ for developing web applications. The skill is automatically assigned based on the number of mashups a designer owns (relation Γ), according to a threshold-based scale: 1.0 for `expert`, 0.8 for `high confidence`, 0.5 for `medium confidence`, 0.3 for `low confidence` and 0.0 for `unexperienced`. More sophisticated skill update functions will be investigated as future work.

A designer may assign two kinds of information about a Web API in the context of a mashup where it has been used: semantics tags and a rating score. Semantic tags are assigned with the support of the WordNet lexical system. A semantic tag in $\mathcal{S}$ is a triplet $t := \langle n_t, syn, d_t \rangle$, where: (i) n_t is the name of the tag itself; (ii) syn is the set of all the synonyms of t, extracted from WordNet; (iii) d_t is the human readable definition associated with the synset. The rating score μ is assigned by the designer according to the NHLBI 9-point Scoring System. See [3] for more details about semantic tags and rating score assignment.

Example. Let consider the mashup `Imaginalaxy`, tagged by an expert designer $d_1 \in \mathcal{D}$ and composed of the `Amazon S3`, `MySpace` and `Flickr` APIs. Now consider a fictious mashup `MyNewMashup`, developed and tagged by a medium-skilled designer $d_2 \in \mathcal{D}$, who used the `Amazon S3`, `MySpace` and `Facebook` APIs. The two mashups share the `Amazon S3` and `MySpace` APIs, while differ in the choice of the third API[1]. In Figure 1(a) we reported the semantic tags of the `Flickr` and

[1] In this paper, we adopt a toy example. In a real case scenario, all the mashups in the repository which are composed of, among the others, the `Flickr` and `Facebook` APIs should be considered for Web API search and ranking.



Tag name	Synonyms	Definition
sharing	{ }	using or enjoying something with others
picture	{photograph,exposure,photo,pic}	a representation of a person or scene in the form of a print or transparent slide or in digital format
snapshot	{snap,shot}	an informal photograph; usually made with a small hand-held camera

(c)

Fig. 1. A graphical representation of the semantic tag assignment (a) and rating of Web APIs within mashups (b) for the running example; (c) WordNet-based semantic tags definition

`Facebook` APIs assigned by d_1 and d_2, respectively. The expert designer tagged the `Flickr` API with `sharing` and `picture`, while d_2 tagged the `Facebook` API with `sharing` and `snapshot`. The definition of the semantic tags based on WordNet is given in Figure 1(c). Moreover, the expert designer assigned a rating score equal to 0.7 to the `Flickr` API used in the `Imaginalaxy` mashup, while d_2 assigned a higher score (0.9) to `Facebook` API used in `MyNewMashup`, as shown in Figure 1(b).

2.1 Web API Request and Search Targets

A Web API request is formulated depending on the search target the web designer is pursuing. A general definition of Web API request is the following:

$$\mathcal{R} = \langle C_{\mathcal{R}}, \{t_{\mathcal{R}}\}, \{\mathcal{W}_R\}, \mathcal{F}_{\mathcal{R}} \rangle \tag{1}$$

where: (i) $C_{\mathcal{R}}$ is the set of required Web API categories, to be matched against the categories $C_{\mathcal{W}}$ of each available Web API $\mathcal{W} \in \Omega$ in the repository $\Re$; (ii) $\{t_{\mathcal{R}}\}$ is a set of semantic tags specified for the Web API to search for (for each Web API $\mathcal{W} \in \Omega$, $\{t_{\mathcal{R}}\}$ is matched against the semantic tags assigned by the designers to $\mathcal{W}$ for each mashup where the Web API has been used); (iii) $\{\mathcal{W}_R\}$ is an optional set of Web APIs already included in a mashup that the designer aims at completing (if any) (for each Web API $\mathcal{W} \in \Omega$, $\{\mathcal{W}_R\}$ is matched against the Web APIs which compose the mashups where both $\{\mathcal{W}_R\}$ and $\mathcal{W}$ have been included); (iv) $\mathcal{F}_{\mathcal{R}}$ is an optional set of required technical features, among protocols, data formats and security features.

Not all the elements listed in Equation (1) are mandatory. Their specification and use within $\mathcal{R}$ depend on the *search target* pursued by the designer. We identified the following kinds of search targets:

- *single Web API selection*, when the designer is developing a new mashup and aims at finding a Web API by specifying desired Web API categories $C_{\mathcal{R}}$ and semantic tags $\{t_{\mathcal{R}}\}$; optionally, the designer may also specify the desired technical features $\mathcal{F}_{\mathcal{R}}$ for the Web API to search for;
- *mashup completion*, when a mashup is already available and the designer desires to add a new Web API; in this case, $\{\mathcal{W}_R\}$ is the set of Web APIs already included in the mashup under construction and the goal is to receive suggestions about other Web APIs $\mathcal{W}\in\Omega$ which have been used with $\{\mathcal{W}_R\}$ in the same mashups; moreover, if the designer does not explicitly specify $\mathcal{F}_{\mathcal{R}}$, it is automatically composed of the intersections of protocols, data formats and security features of the Web APIs in $\{\mathcal{W}_R\}$, respectively; in fact, in this specific case, the best solution is that all the Web APIs within the mashup share the same protocol, data format and security features;
- *proactive mashup completion*, it is a variant of the previous search target; in this case, the designer does not specify the categories $C_{\mathcal{R}}$ and the semantic tags $\{t_{\mathcal{R}}\}$ of the Web API to search for, but relies on suggestions of the system, which proposes candidate Web APIs based on collective knowledge coming from other existing mashups, which contain Web APIs close to $\{\mathcal{W}_{\mathcal{R}}\}$ and presenting technical features similar to $\mathcal{F}_{\mathcal{R}}$;
- *Web API substitution*, when the designer desires to substitute a Web API in an existing mashup; in this case, $C_{\mathcal{R}}$ and $\{t_{\mathcal{R}}\}$ are specified for the Web API to substitute; the construction of $\mathcal{F}_{\mathcal{R}}$ follows the same procedure of the (proactive) mashup completion.

3 Web API Search and Ranking Metrics

Web API search is performed through computation of similarity metrics. Similarity metrics are applied by combining the information available for Web APIs, namely their categories, technical features, semantic tags and mashups where the Web APIs have been included, considering the particular folksonomy-style model described in the previous section. The overall similarity measure between the request $\mathcal{R}$ and each API $\mathcal{W}\in\Omega$ in the repository $\Re$, denoted with $Sim(\mathcal{R},\mathcal{W})$, is computed as the following linear combination:

$$\begin{aligned} Sim(\mathcal{R},\mathcal{W}) = \omega_1 \cdot Sim_c(\mathcal{R},\mathcal{W}) \ + \ \omega_2 \cdot Sim_t(\mathcal{R},\mathcal{W}) + \\ \omega_3 \cdot Sim_{comp}(\mathcal{R},\mathcal{W}) \ + \ \sum_{i=4}^{N} \omega_i \cdot \overline{Sim_i}(\mathcal{R},\mathcal{W}) \in [0,1] \end{aligned} \tag{2}$$

where $0\leq\omega_i\leq1$ and $\sum_{i=1}^{N}\omega_i = 1$ are weights to be set according to the search target, as summarized in Table 1, where the category similarity is weighted less than the other factors, since the category is only a coarse-grained entry point to look for Web APIs in the `ProgrammableWeb` repository. Specifically:

- $Sim_c(\mathcal{R},\mathcal{W})\in[0,1]$ is the similarity between the requested category and the category of $\mathcal{W}$;

Table 1. The setup of Web API similarity weights depending on the search target

Search target	Request formulation
Single Web API selection	$\omega_1, \omega_2, \omega_i (i = 4..N)$ equally weighted or $\omega_1 = 0.2$ and $\omega_2 = 0.8$ if $\mathcal{F}_\mathcal{R} = \emptyset$
Mashup completion	$\omega_1 = 0.1, \omega_2 = \omega_3 = \omega_i (i = 4..N)$
Proactive mashup completion	$\omega_3 = \omega_i (i = 4..N)$ or $\omega_3 = 1.0$ if $\mathcal{F}_\mathcal{R} = \emptyset$
Web API substitution	$\omega_1 = 0.1, \omega_2 = \omega_3 = \omega_i (i = 1..N)$

- $Sim_t(\mathcal{R}, \mathcal{W}) \in [0, 1]$ is the similarity according to semantic tags;
- $Sim_{comp}(\mathcal{R}, \mathcal{W}) \in [0, 1]$ is the combination of composition similarities between the mashup which contains the set of Web APIs $\{\mathcal{W}_R\}$, where the required one will be included, and each mashup which contains $\mathcal{W}$;
- $\overline{Sim_i}(\mathcal{R}, \mathcal{W}) \in [0, 1]$ is the similarity between $\mathcal{R}$ and $\mathcal{W}$ based on the i-th technical feature, such as protocols or data formats.

For what concerns $Sim_c(\cdot)$ and $Sim_t(\cdot)$ computation, we refer to the details provided in [3]. Here we will focus on the contributions of this paper on mashup composition similarity and Web API similarity based on technical features.

The mashup composition similarity between a mashup m_1 and another mashup m_2, denoted with $MashupSim(m_1, m_2)$, evaluates the number of common Web APIs in the two mashups, that is:

$$MashupSim(m_1, m_2) = \frac{2 \cdot |m_1 \cap m_2|}{|m_1| + |m_2|} \in [0, 1] \tag{3}$$

where $|m_1 \cap m_2|$ denotes the number of common Web APIs in the two mashups, while $|m_i|$ denotes the number of Web APIs in the i-th mashup. To compute the *mashup-based similarity* between the request $\mathcal{R}$ and each available Web API $\mathcal{W} \in \Omega$ in the repository $\Re$, denoted with $Sim_{comp}(\mathcal{R}, \mathcal{W})$, it should be considered that each mashup, where $\mathcal{W}$ has been included together with $\{\mathcal{W}_\mathcal{R}\}$, is developed by a designer $d_i \in \mathcal{D}$, who is characterized by a skill σ_i in web application development. Therefore, similarity with mashups developed by more expert designers should be weighted more than similarity with mashups developed by less expert designers. Therefore, the mashup-based Web API similarity is computed as follows:

$$Sim_{comp}(\mathcal{R}, \mathcal{W}) = \frac{1}{|\mathcal{D}_\mathcal{W}|} \cdot \sum_{i=1}^{|\mathcal{D}_\mathcal{W}|} \frac{\sum_{k=1}^{|M_i|} \sigma_i \cdot MashupSim(\{\mathcal{W}_\mathcal{R}\}, m_i^k)}{|M_i|} \in [0, 1] \tag{4}$$

where: (i) $\mathcal{D}_\mathcal{W} \subseteq \mathcal{D}$ is the set of designers who used the Web API $\mathcal{W}$ in one of their mashups and $|\cdot|$ denotes the set cardinality (that is, the number of designers); (ii) M_i is the set of mashups owned by a designer $d_i \in \mathcal{D}_\mathcal{W}$ and $m_i^k \in M_i$; (iii) σ_i is the skill associated to the designer $d_i \in \mathcal{D}_\mathcal{W}$.

The similarity between the request $\mathcal{R}$ and each Web API $\mathcal{W} \in \Omega$ in the repository $\Re$, based on the i-th technical feature, denoted here as $\overline{Sim_i}(\mathcal{R}, \mathcal{W})$, is defined as follows:

$$\overline{Sim_i}(\mathcal{R},\mathcal{W}) = \begin{cases} Sim_i(\mathcal{R},\mathcal{W}) & for\ single\ Web\ API\ selection \\ Sim_i^{asym}(\mathcal{R},\mathcal{W})\ otherwise \end{cases} \quad (5)$$

where $Sim_i(\mathcal{R},\mathcal{W})$ is computed as the number of common elements for the i-th feature (for instance, the number of common protocols), denoted with $|\mathcal{F}^i_{\mathcal{R}} \cap \mathcal{F}^i_{\mathcal{W}}|$, with respect to the overall number of allowed values, that is:

$$Sim_i(\mathcal{R},\mathcal{W}) = \frac{2 \cdot |\mathcal{F}^i_{\mathcal{R}} \cap \mathcal{F}^i_{\mathcal{W}}|}{|\mathcal{F}^i_{\mathcal{R}}| + |\mathcal{F}^i_{\mathcal{W}}|} \in [0,1] \quad (6)$$

Note that, in this case, common values might mean also compatible values (such as RSS and Atom formats, which are both based on XML). Nevertheless, in our framework, the formula shown in Equation (6) is applied only for single Web API selection. When a (proactive) mashup completion or a Web API substitution must be performed, we provide an asymmetric variant of this formula, that assigns more relevance to the values of the i-th feature required in the request $\mathcal{R}$, that is:

$$Sim_i^{asym}(\mathcal{R},\mathcal{W}) = \frac{|\mathcal{F}^i_{\mathcal{R}} \cap \mathcal{F}^i_{\mathcal{W}}|}{|\mathcal{F}^i_{\mathcal{R}}|} \in [0,1] \quad (7)$$

This variant has been designed to take the viewpoint of the Web API request in terms of required technical features: if the Web API $\mathcal{W}$ has been designed considering all the required values for a feature, this is considered as the best situation, no matter $\mathcal{W}$ allows additional values for the same feature. This is due to the specific nature of mashup completion and Web API substitution search targets. In fact, in these cases the required feature values are the ones of other Web APIs already included in the mashup under development. Therefore, the idea is to assign more relevance to those Web APIs that share (compatible) features with the other Web APIs already in the mashup to be developed.

3.1 Ranking Metrics for Web API Search

The Web APIs returned as search results among the ones available in the repository $\Re$ (which we denote with $\{\mathcal{W}'\} \subseteq \Omega$) are those whose overall similarity is equal or greater than a threshold $\gamma \in [0,1]$ set by the web designer. Search results in $\{\mathcal{W}'\}$ are ranked according to the values of $Sim(\mathcal{R},\mathcal{W})$. Nevertheless, additional ranking criteria can be added based on the ratings assigned by web designers to Web APIs. Formally, search results in $\{\mathcal{W}'\}$ are ranked according to the following equation:

$$\rho(\mathcal{W}') = Sim(\mathcal{R},\mathcal{W}) \cdot \frac{1}{|\mathcal{D}_{\mathcal{W}'}|} \cdot \sum_{i=1}^{|\mathcal{D}_{\mathcal{W}'}|} \frac{\sum_{k=1}^{|M_i|} \sigma_i \cdot \mu(\mathcal{W}', m_i^k, d_i)}{|M_i|} \in [0,1] \quad (8)$$

where: (i) $\mathcal{D}_{\mathcal{W}'}$ is the set of the designers who rated the Web API $\mathcal{W}'$ in some mashups and $|\mathcal{D}_{\mathcal{W}'}|$ denotes the cardinality of the set (that is, the number of designers); (ii) for each designer $d_i \in \mathcal{D}_{\mathcal{W}'}$, M_i represents the set of mashups where

d_i used (and rated) the Web API $\mathcal{W}'$ and $m_i^k \in M_i$; (iii) the ratings assigned by more expert designers (that is, designers who present higher σ_i) are considered as more important.

4 Conclusions

In this paper we proposed a framework which implements a set of techniques and mechanisms to support Web API search and ranking by exploiting the "wisdom" of the other designers who used the Web APIs for their own mashups. Designers' wisdom is properly weighted using their expertise in developing web mashups and semantic tags and ratings they assigned on Web APIs. As future extensions of this work, also social relationships between developers could be used to automatically infer useful information, such as developers' reliability, while a complete implementation and in-depth evaluation of the metrics is being developed.

References

1. Gruber, T.: Collective knowledge systems: Where the Social Web meets the Semantic Web. Journal Web Semantics: Science, Services and Agents on the World Wide Web 6, 4–13 (2008)
2. Torres, R., Tapia, B., Astudillo, H.: Improving Web API Discovery by leveraging social information. In: Proceedings of the IEEE International Conference on Web Services, pp. 744–745 (2011)
3. Bianchini, D., De Antonellis, V., Melchiori, M.: Semantic Collaborative Tagging for Web APIs Sharing and Reuse. In: Brambilla, M., Tokuda, T., Tolksdorf, R. (eds.) ICWE 2012. LNCS, vol. 7387, pp. 76–90. Springer, Heidelberg (2012)
4. Gomadam, K., Ranabahu, A., Nagarajan, M., Sheth, A., Verma, K.: A Faceted Classification Based Approach to Search and Rank Web APIs. In: Proc. of International Conference on Web Services (ICWS), pp. 177–184 (2008)
5. Bianchini, D., De Antonellis, V., Melchiori, M.: A Linked Data Perspective for Effective Exploration of Web APIs Repositories. In: Daniel, F., Dolog, P., Li, Q. (eds.) ICWE 2013. LNCS, vol. 7977, pp. 506–509. Springer, Heidelberg (2013)
6. Bianchini, D., De Antonellis, V., Melchiori, M.: A Multi-perspective Framework for Web API Search in Enterprise Mashup Design. In: Salinesi, C., Norrie, M.C., Pastor, Ó. (eds.) CAiSE 2013. LNCS, vol. 7908, pp. 353–368. Springer, Heidelberg (2013)
7. Shafiq, M., Alhajj, A., Rokne, J.: On the social aspects of personalized ranking for web services. In: Proc. of 13th IEEE Int. Conference on High Performance Computing and Communications, pp. 86–93 (2011)
8. Dojchinovski, M., Kuchar, J., Vitvar, T., Zaremba, M.: Personalised Graph-Based Selection of Web APIs. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 34–48. Springer, Heidelberg (2012)
9. Hotho, A., Jäschke, R., Schmitz, C., Stumme, G.: Information Retrieval in Folksonomies: Search and Ranking. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 411–426. Springer, Heidelberg (2006)