Personal Web API Recommendation Using Network-based Inference

Svetlana Omelkova[1] and Peep Küngas[1]

[1] University of Tartu, Estonia ,
svetlana.omelkova@ut.ee
[2] peep.kungas@ut.ee

Abstract. In this paper, we evaluate a generic network-based inference algorithm for Web API recommendation. Based on experimental data collected from the Programmable Web repository, we construct two tripartite networks: one where the nodes are Web APIs, users and mashups, and another where the nodes are Web APIs, users and tags. Experimental results show that the network-based inference algorithm yields higher precision, ranking quality and personalization score when applied to the second network. This approach also outperforms three existing methods: a global ranking method, a collaborative filtering method and the Programmable Web recommendation tool.

1 Introduction

The uptake of Web APIs (Application Programming Interfaces) allows developers to conveniently expose data and applications on the Web in a reusable manner. Developers can easily create a Web API composition (a.k.a. a *mashup*) that aggregates data and application logic from different sources. The proliferation of Web APIs brings about significant opportunities for developers, but also a burden of choice. For example, in order to enrich an application with machine translation functionality, a developer needs to choose between a wide range of available solutions such as Google Translate, Microsoft Translator, IBM Watson Machine Translation and other less popular but sometimes equally suitable alternatives. This information overload raises the need for specialized recommender systems that automatically suggest appropriate Web APIs in a given context.

Programmable Web (PW)[3] – the largest online repository of Web APIs – provides a keyword-based recommendation tool and allows users to filter Web APIs by category and/or communication protocol. However, these filters are coarse-grained and sometimes return too many options. For example, more than 200 Web APIs are proposed by PW's recommendation tool when feeding it with the keyword "translation". It is a tedious job to browse through this list and to assess the relevance of each Web API for a given need.

The problem of Web API recommendation has attracted significant attention in the research community. Some authors [12, 5] have proposed to cluster Web APIs based on their functional properties and to use these clusters for Web

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

API recommendation. Other authors [4] have proposed to use Web API tags in conjunction with WSDL descriptions to locate Web APIs providing similar functionality. Bianchini et al. [1] have proposed to semantically annotate Web API interfaces with concepts from standard Web Service taxonomies, and to use these semantic annotations to calculate similarity measures between pairs of Web APIs. Other streams of research have addressed the problem of ranking Web APIs by their relevance relative to a given query, taking into account usage and popularity [6, 2]. Collaborative filtering (CF) has also been applied to address the problem of Web API recommendation [16, 9]. In these approaches, recommendations are produced by computing the similarity between users and analyzing the preferences of the top-k most similar users. Other techniques that have been applied to this problem include latent semantic models and matrix factorization [11, 13]. However in these approaches only historical usage data is employed, while the Web API functionality itself is not taken into consideration.

Another popular approach to recommend Web APIs is to construct networks that relate various types of entities associated to a Web API. For example, Cao et al. [3] construct a network where each node is a mashup. Two mashups are connected via an edge if they share Web APIs and/or functionality tags. In a similar vein, Wang et al. [8] construct User-API and User-Tag networks. Network-based inference techniques can then be applied to these networks in order to produce specific Web API or mashup recommendations.

In this paper, we apply a generic Network-based Inference (NBI) algorithm proposed by Zhou et al. [17] to the problem of recommending Web APIs to a given user and in a given context. We evaluate this algorithm with respect to three metrics (ranking quality, top-k precision, and top-k personalization) using networks extracted from the PW repository. We consider two types of tripartite networks: one where the nodes are Web APIs, users and mashups, and another where the nodes are Web APIs, users and tags. We also compare the performance of Zhou's algorithm with respect to three baselines: a global ranking method, a collaborative filtering method, and the keyword-based recommendation method embodied in the PW repository.

The rest of the paper is structured as follows. Section 2 introduces the NBI algorithm of Zhou et al. and its application to Web API recommendation. Section 3 discusses the dataset and the data extraction steps we have applied. Section 4 presents the evaluation results and Section 5 draws conclusions.

2 Network-based Inference

The NBI algorithm by Zhou et al. [17] is based on a network representation of the input data. In its original form, the algorithm takes as input a simple user-item network, which captures historical data about purchases, downloads or endorsements of an item by a user. The algorithm computes an *importance score* for each item relative to the target user to whom the recommendation is being made. Figure 1(a) illustrates this recommendation approach. A blue circle

denotes a target user. Initially, one unit of *resource*[4] is allocated to each item node linked to the target user. Other items get no resource. In the first step, each item evenly spreads its resource to all connected user nodes (Figure 1(b)). In the second step, each user node evenly spreads its resource to its neighboring items (Figure 1(c)). After these two steps, all items not yet linked to the target user are sorted in descending order according to their sum of resource value. Items with the highest recommendation score can now be recommended to the user. Zhou et al. show that this method outperforms a Collaborative Filtering (CF) recommendation method based on user similarity, with respect to recommendation accuracy and computational complexity [17]. Later, Yu et al. [10] conducted a comparative evaluation of different variants of this generic NBI algorithm, which confirmed the applicability and good performance of this approach.

The generic NBI algorithm [17] has been further developed in [15], leading to a more advanced recommendation algorithm that uses the procedure described above for a *user-item-tag* tripartite network. Here, the tags capture information about the preferences of a user and about the contents of an item. The tags are assigned via a collaborative tagging system. The extended algorithm uses the same two-step resource spreading principle outlined above, but now the resource spreads in parallel along the *user-item* and *item-tag* networks. The final resource is computed as a linear superposition of two resource vectors obtained from these parallel spreading processes. The evaluation results show that the use of collaborative tags significantly improves the recommendation accuracy.

In the Web API ecosystem, tags represent an abstract functionality of a Web API rather than personal preferences of a user, since tags are assigned to a Web API by its developer(s). Therefore, Web APIs with identical tags have the same abstract functionality. The fact that two Web APIs are used in the same mashup can also be treated as an indicator of similar functionality. In this article we use either the information captured in the tags or the information about inclusion of a Web API in a mashup to construct two types of tripartite networks.

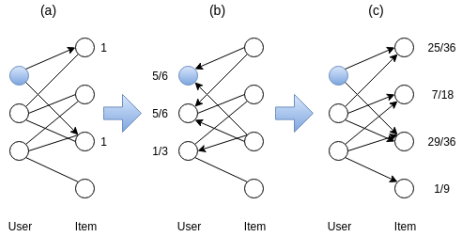

Fig. 1: An illustration of NBI resource spreading procedure.

3 Dataset

For benchmarking we used the PW repository crawled on 22 Oct. 2015. Information about published Web APIs, mashups, and users, who published or followed

[4] In this setting, the resource being flooded into the network is the total recommendation capacity.

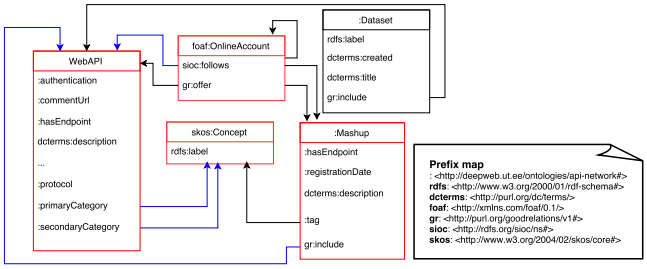

Fig. 2: Dataset structure

at least one Web API or mashup has been collected. All available metadata has been mapped into an RDF graph using the API-Network ontology proposed by us earlier [7]. The structure of the dataset is depicted in Figure 2. The dataset includes a description of itself, descriptions of *WebAPI*s with their data properties such as *authentication*, *commentUrl* and textual *description* as well as object properties capturing relationships with other classes. For example object properties *primaryCategory* and *secondaryCategory* connect *WebAPI* with its categories which are an instances of a *skos:Concept* class. Another member of the dataset is *foaf:OnlineAccount*, which represents a user. For each user, the dataset tells us which mashups and Web APIs the user follows and which ones he/she provides. A user can also follow other users. The *Mashup* class provides detailed information about a given mashup, e.g. *registrationDate* and textual *description*, as well as lists of *WebAPI*s it *include*s and its *tag*s. *Tag* is an RDF property capturing the functionality of a *Mashup*. *Category* on the other hand captures the functionality of a *WebAPI*.

The basic statistics of the dataset are listed in Table 1. The table shows the number of Web API, User, Mashup and Tag nodes in the network as well as the total number of different links between them. The instances of red-colored classes in Figure 2 are nodes and those of blue-colored classes are links.

Number of RDF triples	497734	Earliest published Web API	2005-06-02
Number of Web APIs	14028	Latest published Web API	2015-10-19
Number of Users	70328	Num of User-WebAPI links	121530
Number of Mashups	7692	Num of WebAPI-Mashup links	15758
Number of Categories	461	Num of WebAPI-Tag links	40992

Table 1: Programmable Web dataset's basic counts

4 Evaluation

We evaluate the performance of the NBI method for Web API recommendation with respect to the question "Which Web APIs a given user would most probably follow given his/her past preferences?". For the experiment we extracted

two networks from the PW RDF graph (Section 3). The first one is a tripartite User-API-Tag network where nodes are instances of classes *foaf:OnlineAccount*, *:WebAPI* and *scos:Concept*. A link exists between a User node and a WebAPI node if an RDF property *sioc:follows* exists between these nodes. Similarly, a link exists between a WebAPI node and a Tag node if an RDF property *:primaryCategory* exists between these nodes. The second network is a tripartite User-API-Mashup network, which is identical to the first network except that the third type of node corresponds to class *:Mashup* and a link exists if there is an RDF property *gr:include* between a Mashup and a WebAPI node.

For the evaluation we use three baseline recommendation techniques. The first baseline is a global ranking method (GRM), which sorts all presented Web APIs in descending order by in-degree, i.e. by the number of followers. Although it is a non-personalized method, it is widely used in practice. GRM will always favor the most popular Web APIs such as *facebook*, *google-maps* or *twitter*. The second baseline is CF based on item similarity. The similarity of two items is computed based on the number of users who follow both items. Finally, the third baseline is the Web API recommendation tool provided by the PW repository[5]. The tool provides a keyword-based Web API search method with the option of filtering by category and/or interaction protocol. To customize the PW recommendation tool for our experiment, we determine the set of interests of a target user. To determine the latter, we extracted the list of categories of Web APIs that a target user is already following. From this list of categories of interest, we selected the top-2 most prevalent in the list. These two categories form a query to the PW recommendation tool, and the result is considered to be a personal recommendation list for the target user.

4.1 Evaluation metrics

A good recommendation method is characterized by its ability to identify an item that the target user would like (use, buy). We divide the dataset (121530 User-API edges) into training and test set in the proportion 9/1 respectively. The training set is used as the actual input for the evaluated method. The obtained recommendation list is compared with the test set. Finally, the degree of match between them is quantified using three evaluation metrics: precision, ranking score and personalization score.

The first metric is a *ranking score* (r). For a target user u_i, each method under study ranks Web APIs that have not been already picked by u_i in the training set (uncollected items), by recommendation score in ascending order. The result is an ordered personal recommendation list for target user u_i. For each User-API pair $(u_i - api_j)$ from the test set we take a positional number of api_j in the recommendation list of u_i and normalize this quantity by the length of the recommendation list. This value is the individual ranking score r. A small r means that the api_j from the test set has been ranked highly by the recommendation algorithm, which implies better recommendation accuracy. A

[5] http://www.programmableweb.com/category/all/apis

mean ranking score of the method is computed by averaging individual ranking scores across all User-API pairs in the test set.

The second metric is *precision*. This metric focuses only on top-k positions of the recommendation list since in practice only the top recommended items are viewed by the user. Individual precision is computed as a number of Web APIs from the test set appearing in the top-k positions normalized by k. A mean precision of the method is computed as an average value of individual precisions for all users in the test set.

The third metric is *personalization score*. It measures the difference between the recommendation lists of pairs of distinct users. The difference between two lists can be measured as a Hamming distance. The Hamming distance between two users i and j can be computed using the formula: $D_{ij}(k) = 1 - C_{ij}(k)/k$, where the $C_{ij}(k)$ is the a number of common elements in the top-k positions of the recommendation lists of the two users. The mean Hamming distance is computed by averaging the Hamming distance across all pairs of users. The higher is the mean Hamming distance, the more personalized is the recommendation.

4.2 Experimental results

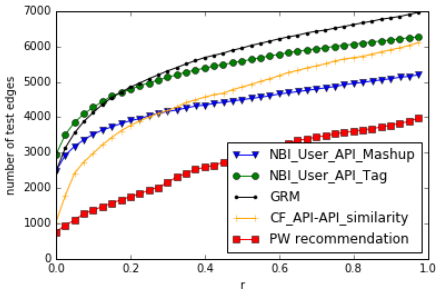

Fig. 3: Cumulative distribution of ranking scores of api_j in the u_i's recommendation list computed by five methods.

Table 2: Evaluation of five recommendation methods using three accuracy metrics (k =30)

Method	r	P(k)	D(k)
NBI User-API-Mashup	0.168	0.0135	0.976
NBI User-API-Tag	0.153	0.0137	0.991
GRM	0.197	0.007	0
CF API-API similarity	0.26	0.004	0.643
PW recommendation	0.335	0.004	0.553

The performance of the described methods are shown in Table 2. For each of the five methods under study we compute the mean ranking score r, mean precision $P(k)$ and mean personalization $D(k)$, for k=30. This latter value is the default number of entries appearing on the first page returned by PW's recommendation tool. A good recommendation method is expected to have a low ranking score, high precision, and high personalization.

Among the two NBI methods, the User-API-Tag method outperforms the User-API-Mashup by achieving a lower ranking score and slightly higher precision and Hamming distance. Therefore, additional usage information (i.e. inclusion of API into mashups) has a lower impact for the recommendation than the tag information. Previous studies show that the PW ecosystem has an unbalanced, long tail distribution of Web API usage frequency [14]: only a tiny portion

of Web APIs is well-known and widely used. The benefits of using tags can also be seen in Figure 3 where the cumulative distribution of individual ranking scores is shown. The figure shows that the NBI User-API-Tag method (green line) outperforms the NBI API-User-Mashup (blue line) in terms of ranking score. Both NBI methods have high personalization scores ($D(k)$).

The GRM appears to be also quite accurate regarding the mean ranking score. This fact can be seen in Figure 3 (black line). The only limitation of GRM is that it has a non-personalized nature. Such good performance of the non-personalized method is possible only on highly imbalanced ecosystems, i.e. the winning strategy is always to recommend the most popular items, no matter what are the personal preferences.

CF based on API-API similarity has relatively high mean ranking score and low precision and personalization scores. The worst performance is that of the PW recommendation tool. On average only 55% of items are different in the top-k recommendation lists for various users (Hamming distance equals to 0.553). Its poor ranking ability is reflected as a red line on Figure 3. Most probably the PW recommendation tool does not take into account how often a recommended Web API has been used in the past. At the same time, such a recommendation strategy can help to overcome the so-called cold start problem, i.e. recently registered Web APIs remain unnoticed by many possible users. By recommending not yet popular Web APIs, the PW recommendation tool helps them to gain popularity, but at the cost of accuracy.

Overall the two NBI methods outperform all three other recommendation methods chosen for evaluation. The NBI methods achieve higher precision and personalization values and also better ranking score.

5 Conclusion

In this paper we studied the applicability of a generic Network-based Inference method to Web API recommendation. We compared two NBI algorithms based on different tripartite networks (User-API-Tag and User-API-Mashup) against three baselines, namely global ranking, collaborative filtering and the recommendation tool provided by the PW repository. We showed that the NBI algorithms, especially the one based on the User-API-Tag network, outperform other methods in terms of ranking score, precision, and personalization.

The main benefit of NBI is that it can work on diverse networks. Different networks can help to answer different questions. For example, an API-Tag bipartite network can be used to recommend additional Web API categories, while a Mashup-API network can be used to supplement a possibly incomplete mashup. And to increase recommendation accuracy, one can extend the input network and make it tripartite by adding another type of node.

Acknowledgment. This research is funded by the Estonian Research Council. We would like to thank Prof. Marlon Dumas for the language editing and proofreading.

References

1. Bianchini, D., De Antonellis, V., Melchiori, M.: A recommendation system for semantic mashup design. In: Database and Expert Systems Applications (DEXA), 2010 Workshop on. pp. 159–163. IEEE (2010)
2. Bianchini, D., De Antonellis, V., Melchiori, M.: Exploiting social tagging in Web API search. In: On the Move to Meaningful Internet Systems: OTM 2013 Conferences. pp. 764–771. Springer (2013)
3. Cao, B., Liu, J., Tang, M., Zheng, Z., Wang, G.: Mashup service recommendation based on user interest and social network. In: Web Services (ICWS), 2013 IEEE 20th International Conference on. pp. 99–106. IEEE (2013)
4. Chen, L., Wang, Y., Yu, Q., Zheng, Z., Wu, J.: WT-LDA: user tagging augmented LDA for web service clustering. In: Service-Oriented Computing, pp. 162–176. Springer (2013)
5. Elgazzar, K., Hassan, A.E., Martin, P.: Clustering WSDL documents to bootstrap the discovery of web services. In: 2010 IEEE International Conference on Web Services. pp. 147–154. IEEE (2010)
6. Gomadam, K., Ranabahu, A., Nagarajan, M., Sheth, A.P., Verma, K.: A faceted classification based approach to search and rank Web APIs. In: Web Services, 2008. ICWS'08. IEEE International Conference on. pp. 177–184. IEEE (2008)
7. Omelkova, S., Küngas, P.: A linked data model for Web API-s. In: Perspectives in Business Informatics Research - 14th International Conference, BIR 2015, Tartu, Estonia, August 26-28, 2015, Proceedings. pp. 48–63 (2015)
8. Wang, J., Chen, H., Zhang, Y.: Mining user behavior pattern in mashup community. In: Information Reuse & Integration, 2009. IRI'09. IEEE International Conference on. pp. 126–131. IEEE (2009)
9. Yao, L., Sheng, Q.Z., Segev, A., Yu, J.: Recommending web services via combining collaborative filtering with content-based features. In: Web Services (ICWS), 2013 IEEE 20th International Conference on. pp. 42–49. IEEE (2013)
10. Yu, F., Zeng, A., Gillard, S., Medo, M.: Network-based recommendation algorithms: A review. arXiv preprint arXiv:1511.06252 (2015)
11. Yu, Q.: Decision tree learning from incomplete QoS to bootstrap service recommendation. In: Web Services (ICWS), 2012 IEEE 19th International Conference on. pp. 194–201. IEEE (2012)
12. Yu, Q., Rege, M.: On service community learning: A co-clustering approach. In: Web Services (ICWS), 2010 IEEE International Conference on. pp. 283–290. IEEE (2010)
13. Yu, Q., Zheng, Z., Wang, H.: Trace norm regularized matrix factorization for service recommendation. In: Web Services (ICWS), 2013 IEEE 20th International Conference on. pp. 34–41. IEEE (2013)
14. Yu, S., Woodard, C.J.: Innovation in the programmable web: Characterizing the mashup ecosystem. In: Service-Oriented Computing–ICSOC 2008 Workshops. pp. 136–147. Springer (2009)
15. Zhang, Z.K., Zhou, T., Zhang, Y.C.: Personalized recommendation via integrated diffusion on user–item–tag tripartite graphs. Physica A: Statistical Mechanics and its Applications 389(1), 179–186 (2010)
16. Zheng, Z., Ma, H., Lyu, M.R., King, I.: Wsrec: A collaborative filtering based web service recommender system. In: Web Services, 2009. ICWS 2009. IEEE International Conference on. pp. 437–444. IEEE (2009)
17. Zhou, T., Ren, J., Medo, M., Zhang, Y.C.: Bipartite network projection and personal recommendation. Physical Review E 76(4), 046115 (2007)