IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON COLLABORATION FOR INTERNET OF THINGS

Received December 30, 2018, accepted January 9, 2019, date of publication January 24, 2019, date of current version February 8, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2894297

Mining Collaboration Patterns Between APIs for Mashup Creation in Web of Things

MINGDONG TANG[1,2], YANMIN XIA[2], BING TANG[2], YONGMEI ZHOU[1], BUQING CAO[2], AND RONG HU[2]

[1]School of Information Science and Technology, Guangdong University of Foreign Studies, Guangzhou 510006, China
[2]School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411201, China

Corresponding author: Yongmei Zhou (yongmeizhou@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61572186, Grant 61602169, Grant 61702181, and Grant 61873316, in part by the Natural Science Foundation of Hunan Province under Grant 2018JJ2135, and in part by the Scientific Research Fund of Hunan Provincial Education Department under Grant 16C0643.

ABSTRACT The Web of Things (WoT) extends the concept of ''Internet of Things (IoT)'' in that smart devices in the physical world can be interacted with or integrated via popular web technologies (e.g., HTML, HTTP, and Web API). With the WoT, smart devices can use Web APIs to make their data or functionalities accessible by software. With the popularization of Web 2.0 Mashup applications, creating Mashup applications for the IoT (or WoT) via combining different APIs, also has aroused increasing interests. This paper proposes an approach to mining collaboration patterns between APIs to aid mashup creation for the WoT. The goal of the approach is to disclose what kinds of Web APIs are frequently combined together in mashup creation and what kinds of API combination are popular. Based on a real-world mashup and Web API repository, PragrammableWeb.com, we exploit the text description and tags of Web APIs and employ an FP-growth-based association mining algorithm to discover popular collaboration patterns between APIs. To overcome the deficiency caused by tag sparsity, the approach also develops a method based on TF/IDF to expand the tags of Web APIs. The experimental results validated the performance of the proposed approach.

INDEX TERMS Web API, collaboration patterns, association rule mining, web of things, mashup, tag expansion.

I. INTRODUCTION

Internet of Things (IoT) involves extending Internet connectivity beyond standard devices, such as desktops, laptops, smartphones and tablets, to any range of traditionally dumb or non-internet-enabled physical devices and everyday objects. With the IoT, a brand new world of possible applications is unveiled. However, due to the lack of standards, even development of a simple application involving the heterogeneous devices of the IoT is a non-trivial task and still requires extensive skills and time [1]. It becomes highly desirable that developers should be able to quickly build IoT applications only by recombining ready-made building blocks, just like using APIs on personal computers or Internet.

To make the smart devices or objects in the IoT more accessible and integrable, the concept ''Web of Things'' (WoT) is developed [2]. Similar to what the Web (Application Layer) is to the Internet (Network Layer), WoT provides an Application Layer that simplifies the creation of IoT applications. With the WoT, smart devices in the physical world can be easily interacted with or integrated via popular Web technologies, such as HTML(Hyper Text Mark-up Language), HTTP(HyperText Transfer Protocol) and REST(Representational State Transfer), and thus all devices can be integrated into the existing Web. According to the WoT architecture [3]–[5], all smart things should be exposing their data or services through RESTful APIs. RESTful APIs refers to Web APIs or Web services that conform to the REST architectural style, in which data and functionality are considered Web resources and are accessed using Uniform Resource Identifiers (URIs). With WoT, it is expected that applications can be quickly developed via combining the APIs of different smart things.

With the prevalence of Web 2.0 mashup applications, creating mashup applications for IoT(or WoT) via combining different APIs, also has aroused great and increasing interests. A number of WoT mashup applications and tools have been created and are being actively developed such as [6] and [7]. For example, the Japan Geigermap

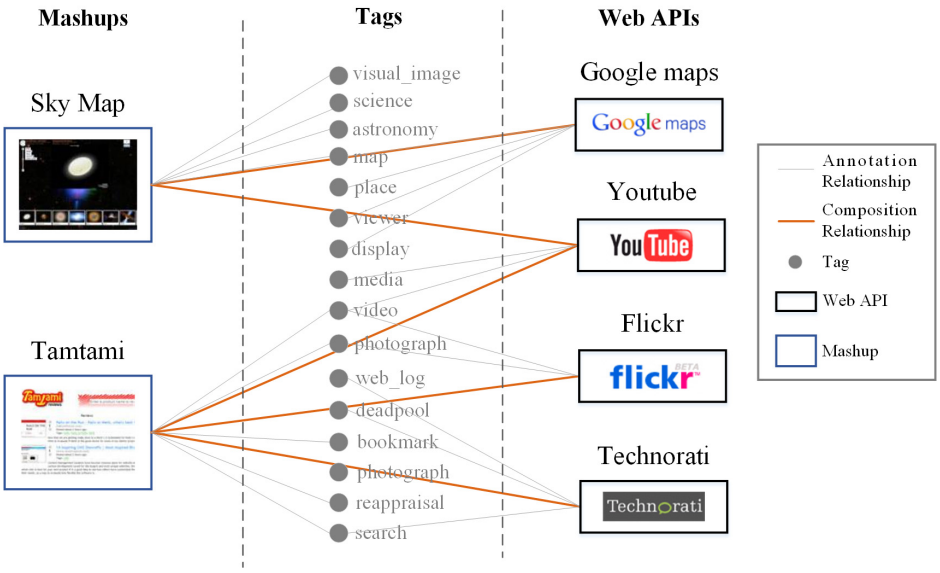


FIGURE 1. Relationships among Web APIs, mashups and tags.

(http://japan.failedrobot.com), which is a WoT mashup application developed by integrating Google Map APIs and geiger counter readings, can visualize the crowd-sourced radiation geiger counter readings across Japan on a Google map. Another representative example is the bike sharing applications (such as Ofo and Mobike), which integrate the shared bikes to the Internet and Web via combining GPS, map APIs and payment APIs. With the advance of IoT and WoT, it is expected that the development of the APIs and mashup applications for IoT and WoT would be greatly accelerated.

Though mashuping Web APIs makes the application creation easier to developers, how to identify suitable API combinations is still a challenge issue for the huge number of APIs on the Web. To aid mashup creation and API combination discovery for WoT, this paper proposes an approach to mining popular collaboration patterns between APIs. The collaboration patterns are mined to reveal what kinds of Web APIs are frequently combined in mashup creation and what functionalities of Web APIs are popular in mashup creation. Based on the observation that most Web APIs and their mashup applications have user-added tags for annotating their functionalities and application scenarios, the proposed approach exploits the tags of Web APIs and mashups to mining API collaboration patterns. FIGURE 1 shows the associations among Web APIs, mashups and tags. As we can see, a mashup is likely to comprise multiple Web APIs, e.g., the mashup "Sky Map" is composed of "Google maps" and "Youtube". And a mashup or a Web API is likely to have a few user-annotated tags, e.g., "Youtube" is tagged with "video" and "media", which means that the functions of "Youtube" API are related to "video" and "media". If two tags frequently co-occur in API combinations in mashup creation, we suppose that there is a strong association between the functionalities represented by them, and view them as an API collaboration pattern. To summarize, in this paper we make the following contributions.

1) We propose a Web API tag association mining algorithm based on FP-growth, to obtain the co-occurrence relationships among tags in API combinations.
2) We employ a set of text processing techniques (such as the Stanford CoreNLP, WordNet and TF-IDF) to tackle the inconsistency and sparsity of tags, so as to improve the quality of the API collaboration pattern mining.
3) We conduct a set of extensive experiments using a real-world mashup and Web API repository, ProgrammableWeb.com. The experimental results show the proposed approach achieves a significant improvement in terms of precision and F-Measure, compared with the other comparative approaches.

The rest of this paper is organized as follows: Section 2 surveys related work on Web API association mining. Section 3 presents the approach to mining Web API collaboration patterns based on API tag association rule mining, namely WACP. Section 4 discusses the experimental settings and results. Finally, Section 5 concludes this paper and outlines future work.

II. RELATED WORK

Web APIs or Web services have been widely used for developing applications in various application scenarios such as business process management [8], scientific workflow management [9], [10], and wireless sensor networks [11], [12]. With the prevalence of Web 2.0, there is a great need for common users without much programming skills to create their personal applications. Against this background, various mashup tools emerge, and mashuping up different Web APIs to create applications has become very popular. However, with the huge number of APIs on the Web, it is still a non-trivial work to find appropriate APIs for mashup creation. A number of research studies have exploited the historical usage information of Web APIs to help mashup creation, and they are surveyed as follows.

Some studies focus on mining the collaboration relations between APIs and incorporate them into API recommendation for mashup creation. For example, [13] explores the co-invocation history between Web APIs and propose a latent model to capture the underlying relations of APIs in mashup creation. Reference [14] recommends APIs for mashups by integrating functional and social relations between APIs, mashups and users by a coupled matrix factorization model. Reference [15] takes the categories of APIs into considerations to recommend APIs for mashup creation by first categorizing APIs by functionality and ranking APIs within each category. However, all of above methods aim at exploiting API collaboration relationships to recommend APIs for mashup creation. This paper focuses on mining API collaboration patterns, instead of particular collaboration relationships.

A few studies have been conducted to mine collaboration or mashup patterns of Web APIs. Goarany *et al.* [16] proposed an approach based on the tags of mashups and Web APIs to predict mashup patterns of APIs. However, it can only generate 2-item association rules and did not address the shortcomings of user annotated tags such as inconsistencies, typos and proliferation of synonyms. Ni *et al.* [17] proposed a rule-based decision tree algorithm to mine both positive and negative tag collaboration rules from the annotated tags of Web services, and then applied them to service recommendation. Again, the inconsistencies, typos and proliferation of synonyms of tags were not taken into account and only 2-item association rules were generated. Gao *et al.* [18] proposed a Service Co-occurrence LDA (SeCo-LDA) model that mines latent topic models over service co-occurrence patterns. However, it's not easy to use a few words to describe the topic of a service co-occurrence pattern, and thus making the service co-occurrence patterns hard to understand. Reference [19] investigated the evolution of service composition patterns based on SeCo-LDA.

There are also several studies which employed a network-based approach to mining collaboration or mashup patterns between Web APIs. Han *et al.* [20] mined the integration patterns of Web APIs or mashups in the Programmable ecosystem via using the tag-tag network, which was built from the tag co-occurrence of Web APIs in mashups. Tang *et al.* [21] also developed a tag pair network, which is built from the annotated tags of Web services and the Web service collaboration network. Popular tag pairs were identified in their work.

Different from previous work, this work fixes the defects of user annotated tags such as inconsistencies, typos and proliferation of synonyms, and can generate association rules other than 2-itemsets. We unify the different forms of every tag and consolidate all synonyms of tags. We also extract keywords from the Web API description documents to expand the tag sets of Web APIs, so that the data sparsity issue of tags is relieved. Finally, we employ a FP-growth-based algorithm to mine tag associations between Web APIs and use the strongest tag association rules to disclose Web API collaboration patterns.

III. THE APPROACH

In this section, we firstly overview the proposed approach WACP. The overall process and components of WACP are briefly introduced. Then every procedure of WACP, such as tag expansion, tag cleaning and processing, association rule mining and API collaboration pattern discovering, are described in detail.

A. OVERVIEW

FIGURE 2 presents the overall process of the proposed approach WACP. As it shows, WACP has the following major procedures:

1) Data collection: WACP depends on the tags of Web APIs and the historical collaboration information of Web APIs in mashup creation. Therefore, it is important to obtain such data. Fortunately, these data can be found through popular Web API portals or search engines such as ProgrammableWeb.com.
2) Tag expansion: Since an API usually has only a few tags in the original dataset, which may be insufficient to depict the functionalities of the API, we use this procedure to generate more tags for APIs. To do this, we extract keywords from the Web API description documents and use TF-IDF to identify important keywords. The top-ranked keyworks are used to expand the tag set for each API.
3) Tag processing: In this procedure, we perform word lemmatization and synonym consolidation for the tags of Web APIs, to ensure the unambiguity and accuracy of tag expressions. We also generate a tag set for each mashup, which is treated as a transaction in this work. The tag set is the union of the tags of all APIs that co-occurred in the mashup.
4) Tag association mining: In this procedure we employ a FP-growth-based algorithm to analyze the tag set of Web APIs, mine all association rules among the tags and calculate their support and confidence degrees. The strong association rules with support and confidence greater than given thresholds are identified.
5) API collaboration pattern generation: In this procedure we filter the tag association rules mined in the above step to remove meaningless association rules and the remained strong association rules are selected and used as Web API collaboration patterns, which disclose what functionalities or characteristics of Web APIs make more sense in mashup creation.

B. TAG EXPANSION

We employed a TF-IDF-based method to mine the description documents of APIs to expand the tag sets of APIs. Term frequency-inverse document frequency (TF-IDF) [22] is a numerical statistic method that is used to evaluate how

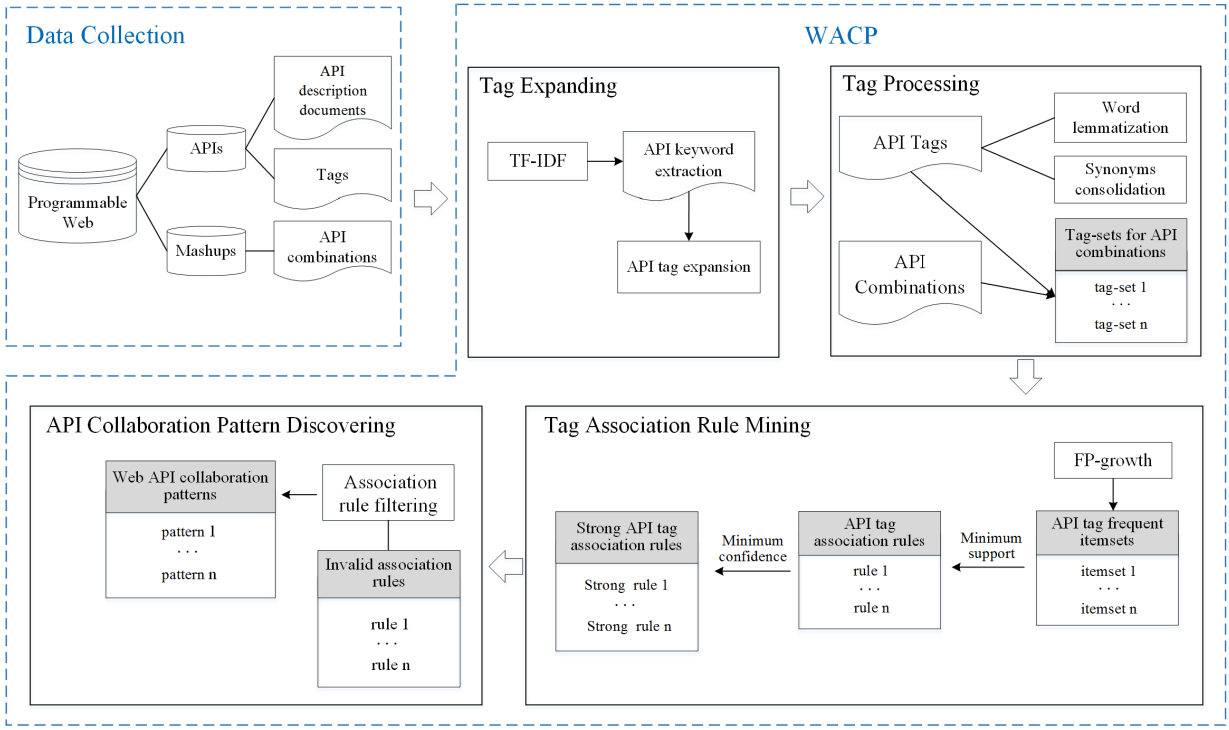


FIGURE 2. The overall process of WACP.

important a word is to a document. The related definitions about TF-IDF are listed as follows:

Definition 1: TF (Term Frequency), which refers to the frequency at which a given word appears in a given document. Its calculation formula is:

$$TF_{x,y} = \frac{n_{x,y}}{\sum_k n_{k,y}} \tag{1}$$

where $n_{x,y}$ is the number of times that term x occurs in document y, $\sum_k n_{k,y}$ is the total number of terms (k) included by the document y.

Definition 2: IDF (Inverse Document Frequency), is a measure of how much information a word or term provides, that is, whether the term is common or rare across all documents. It is calculated using the following formula:

$$IDF_x = \log \frac{N}{n_x + 1} \tag{2}$$

where N represents the number of all documents, and n_x represents the number of documents where the term x appears. The denominator $n_x + 1$ in Eq. (2) is used to avoid division-by-zero when the term is not in the corpus of all documents. The logarithm is used to narrow the range of IDF, since the number of documents may be very high. The smaller is the number of documents that contains term x, the larger is the IDF value of the term x.

To determine whether a term should be treated as a keyword, we should not only take into account the IDF value, but also consider the TF value, that is the complete TF-IDF. The TF-IDF is calculated as:

$$TF - IDF_{x,y} = TF_{x,y} \times IDF_x \tag{3}$$

This work employs TF-IDF to extract important keywords from the description document of a Web API, and then use them as new tags annotating the API. Before extracting API keywords, we first perform a pre-processing procedure to all Web APIs' description text, via the following two steps:

- **Step 1.** Remove special characters such as stop words and punctuation symbols, such as "a", "the", "+", ",", etc.
- **Step 2.** Word segmentation. Split the compound words into simple words. For example, the word "voicemessages", will be split into two words "voice" and "messages".

After all the description documents of Web APIs are preprocessed, we calculate the TF-IDF value of each word appeared in the Web API description documents, and then sort the words in a descending order according to their values of TF-IDF. Finally. For each Web API, we select the top K (K = 1, 2, 3, ...) keywords with the largest TF-IDF values as new tags to expand the original tag sets of Web APIs.

C. TAG CLEANING AND CONSOLIDATION

After Web API tag expansion, the Web API tags need to be further cleaned and consolidated to handle their expression issues such as inconsistencies and proliferation of synonyms. The steps are as follows:

- **Step 1.** *Word Lemmatization*. The Stanford CoreNLP is used to do lemmatization, due to that there are some Web API tags with different forms are originated from the same word, such as *messaging* and *message*. This inconsistency may generate different tag association rules with identical meanings, such as the rule "messaging→phone" and "message→phone". Actually, the two association rules should be merged as a single rule. In order to make the Web API collaboration pattern mining more accurate, all the tags need to be lemmatized to their original form.
- **Step 2.** *Synonyms Consolidation*. The WordNet dictionary is used to consolidate the synonyms of Web API tags. Due to the different description habits of different developers, different words may be used to express the same meaning when defining the tags of Web APIs. This leads to many tags with the same meaning, e.g., *bicycle* and *bike*, *telephone* and *phone*, *picture* and *photo*, etc. This proliferation of synonyms may lead to a number of similar API tag association rules, e.g., "telephone→webhook" and "phone→webhook", "place→bicycle" and "place→bike", etc. It is unnecessary to treat these similar association rules as different rules. Instead, the different synonyms of every tag should be consolidated to improve the effectiveness of Web API composition pattern mining.

D. WEB API TAG ASSOCIATION MINING

Through tag expansion, lemmatization and synonyms consolidation, a set of informative and cleaned tags are obtained for each Web API. In this section, we describe how to mine association rules among tags of Web APIs.

Association rule mining [23] is a rule-based machine learning method for discovering interesting relations between variables in large databases. The related definitions are listed as follows:

Definition 3: Let $I = \{i_1, i_2, \cdots, i_n\}$ be the set of all items, and $D = \{t_1, t_2, \cdots, t_m\}$ be the set of all transactions, and each transaction in I has a unique transaction ID and contains a subset of the items in I. A rule is defined as an implication of the form $X \rightarrow Y$, and $X \subseteq I$, $Y \subseteq I$, $X \neq \emptyset$, $Y \neq \emptyset$, $X \cap Y \neq \emptyset$.

Definition 4: Let X be an itemset, Y be another itemset. Support is an indication of how frequently the itemset appears. The support of a rule $X \rightarrow Y$ is defined as the proportion of the transactions in the dataset which contains the itemset X and Y. It is calculated as:

$$support(X \rightarrow Y) = \frac{|t \in D; (X \cup Y \subseteq t)|}{|D|} \tag{4}$$

Definition 5: Let X be an itemset, Y be another itemset, and $X \rightarrow Y$ be an associate rule. Confidence is an indication of how often the rule has been found to be true. The confidence of a rule $X \rightarrow Y$ is defined as the proportion of the transactions that contains X which also contains Y. It is calculated as:

$$confidence(X \rightarrow Y) = \frac{support(X \cup Y)}{support(X)} \tag{5}$$

The problem of mining association rules is to find all rules whose support and confidence are higher than given minimum support and confidence thresholds.

This work exploits the historical API collaboration data in mashup applications to mine tag association rules. That is, we view every mashup as a transaction, and the tags of a transaction (mashup) is the union of the tag sets of the APIs co-occurred in the mashup.

We employ a FP-growth-based algorithm [24] to mine Web API tag associate rules, such as $[tag_{i1}, tag_{i2}, \cdots, tag_{in}] \rightarrow tag_j$. The details of the algorithm are presented in Algorithm 1. It mainly has three phases. The first phase is the FP-tree construction. The second phase is the frequent item mining based on the FP-tree. The third phase is the tag association rule mining.

E. API COLLABORATION PATTERN DISCOVERY

The above algorithm can generate a set of strong tag association rules with support and confidence greater than given thresholds. Among all mined tag association rules, however, some of them may be improper to act as collaboration patterns of Web APIs. For example, for an association rule $[tag_{i1}, tag_{i2}, \cdots, tag_{in}] \rightarrow tag_j$, if $tag_{i1}, tag_{i2}, \cdots, tag_{in}$ and tag_j all have been used to annotate the same API, this rule should not be taken into account for collaboration pattern mining. This is because that these tags belonging to the same API are just used to describe the same API resource, and cannot not reflect the functional association between different Web APIs. Hence, the complete process of mining Web API collaboration patterns has the following three steps:

1) Generate Web API tag association rules based on the FP-growth algorithm, as we present in Algorithm 1.
2) Filter the generated tag association rules to obtain a subset of tag association rules with support and confidence greater than given thresholds.
3) For each rule in the subset of Web API tag association rules generated above, identify whether all tags in it are belonging to the same Web API. If so, remove it from the subset of tag association rules. The remainder tag association rules can be treated as Web API collaboration patterns.

IV. EXPERIMENTS

In this section we firstly describe the dataset for the experimental evaluation of the proposed approach WACP. Then, we present the experimental results for further illustrating the proposed approach. Finally, we evaluate the proposed approach and compare its performance with two baselines.

A. DATASET

The dataset used in this work is crawled from ProgrammableWeb.com, which is the most popular Web API

Algorithm 1 Web API Tag Association Rule Mining Based on FP-Growth

Input: Web API tags, API collaboration database D, minimal support *min_sup*, minimal confidence *min_conf*.
Output: Web API tag association rules.

1. **Phase 1.** FP-tree Construction
   1) Scan D, obtain tag frequent itemset B, then sorted by the support value to get a frequent item list L_{freq}.
   2) Create the FP-tree root node "*Null*", sort all the transactions in D according to L_{freq}.
   3) All items in D are added to FP-tree in turn, and node in the tree contains item and the occurrences of the item. If the added item already exists, the occurrence is incremented by 1. Otherwise, a new node is created, and its occurrence is set to 1, and it is linked to its parent node.
   4) Recursively call this method until all items have been added.
2. **Phase 2.** Mining FP-tree to get frequent items. This step is realized through calling FP-growth(*FP-tree*, *Null*):
3. **Procedure FP-growth**(*Tree*, α):
4. **if** *Tree* contains only single *path* **then**
5. **for** each composition (denoted as β) in *path* **do**
6. generate item $\beta \cup \alpha$, the support value is the *min_sup* of node in β
7. **end for**
8. **else**
9. **for** each tag a_i in the head of *Tree* **do**
10. generate item $\beta = a_i \cup \alpha$, the support value is equal to the support value of a_i
11. construct conditional items and conditional FP-tree (denoted as $Tree_\beta$) for β
12. **if** $Tree_\beta \neq \emptyset$ **then**
13. call FP-growth($Tree_\beta$, β)
14. **end if**
15. **end for**
16. **end if**
17. **Phase 3.** Mining Web API tag association rules
   1) Define the tag association rule as "*condition* $\rightarrow$ *result*".
   2) For all the frequent items in B that satisfy *min_sup*, generate tag association rule, as follows:
18. **Function List<Rule> TagRuleMining**(B)**:**
19. $fprules \leftarrow \emptyset$, $i \leftarrow 0$
20. **if** B is not null **then**
21. **while** $i < B.size$ **do**
22. $result \leftarrow B.get(i)$
23. $condition \leftarrow condition \cup B.subList(0, i)$
24. $condition \leftarrow condition \cup B.subList(i+1, B.size)$
25. generate a rule "*condition* $\rightarrow$ *result*"and put it into the *fprules*
26. i++
27. **end while**
28. **end if**

Algorithm 1 *(Continued.)* Web API Tag Association Rule Mining Based on FP-Growth

29. **return** *fprules* (tag association rules generated by frequent itemset B).
30. 1) Obtain all Web API tag association rules in the form of "*condition* $\rightarrow$ *result*" from *fprules*.
    2) Obtain all strong Web API tag association rules by filtering using a defined *min_conf* threshold.

TABLE 1. Statistical properties of the dataset.

Statistical property	Value
Number of mashups	6970
Number of Web APIs	9135
Number of Web API providers	7242
Number of Web APIs used by all mashups	1194
Number of tags used by all Web APIs	1727
Average number of Web APIs used per mashup	1.31
Average number of tags used per Web API	3.275

TABLE 2. Examples of API tags after tag expansion.

ID	Web API	Tags
1	*websnapr*	image,security,**perform**
2	*beachguard*	beach,search,public,**advisory**
3	*shiny ads*	advertising,**developers**
4	*netdna*	internet,security,**report**
5	*overwolf*	games,social,video,mobile,**client**

and mashup repository on the Web. The dataset is composed of the description information of 9135 Web APIs and 6875 mashups. The description information of each Web API includes its name, tag set and description text, and the description information of each mashup includes its name, tag set and the Web APIs occurred in it. The tags of Web APIs are used for API collaboration pattern mining, while the tags of mashups are used as test data in the experiments.

Table 1 summarizes the statistics of the dataset. As we can see, the total number of tags used by all Web APIs is 1727, and the average number of tags for each API is only about 1727/9135=3.275. This indicates that the tags for annotating Web APIs is very sparse in the dataset. Hence, it is a necessity to expand the tags of APIs via mining more tags from their description text.

B. RESULTS

This section presents some experimental results for the purpose of illustrating the proposed approach WACP.

1) WEB API TAG EXPANSION

As we stated before, we use TF-IDF to extract keywords from the Web API description documents and generate new tags to expand the original tag set for each Web API. Table 2 shows

some examples of Web APIs after tag expansion, where the bold words represent new tags. For example, the Web API *websnapr* has only two tags ''image'' and ''security'' before tag expansion, we use TF-IDF to identify an important keyword ''perform'' in its description text and adopt it as a new tag for the Web API.

2) WEB API TAG PROCESSING

After tag expansion, the expanded tag set are then cleaned and processed through lemmatization, synonyms consolidation, etc. Table 3 shows some examples of API tags after tag cleaning and processing. For example, the Web API *Shiny ads* has two tags ''advertising'' and ''developers'' after tag expansion, as shown in Table 3. After tag cleaning and processing, its two tags are changed to ''advertise'' and ''developer''.

TABLE 3. Examples of API tags after tag processing.

ID	Web API	Tags
1	*websnapr*	image,security,perform
2	*beachguard*	beach,search,populace,advisory
3	*shiny ads*	advertise,developer
4	*netdna*	internet,security,report
5	*overwolf*	game,social,video,mobile,client

3) ASSOCIATION RULE MINING OF WEB API TAGS

As described in Section III, we have used a FP-growth-based association rule mining algorithm to obtain the frequent itemsets of Web API tags based on their co-occurrence in API collaborations. Table 4 shows some examples of frequent itemsets of Web API tags and their frequency occurred in API collaborations.

TABLE 4. Some Examples of web API tag frequent itemsets.

Frequent Itemsets of Web API Tags	Frequency
place,deadpool,photograph	88
map,display,viewer,place,social,location	66
map,event,calendar	62
video,search,web_log	51
display,video	316
map,reference,database,geography	72
display,search,photograph	97
place,photograph,internet	37
viewer,social,webhook	109
map,display,viewer,reference,weather	77

In the experiments, we set the threshold of support degree to 0.007 (50/6875) and the threshold of confidence degree to 0.5, to obtain strong Web API tag association rules. Table 5 shows the top 10 Web API tag association rules for 2-itemsets.

TABLE 5. Top 10 web API tag association rules (only 2-itemsets).

ID	Frequent Itemsets of Web API Tags	Support	Confidence
1	[usa]→tts	0.02	0.77
2	[weather]→display	0.02	0.74
3	[weather]→viewer	0.02	0.74
4	[weather]→place	0.02	0.72
5	[global]→webhook	0.02	0.72
6	[global]→cloud	0.02	0.70
7	[global]→address	0.02	0.70
8	[geolocation]→display	0.01	0.78
9	[geography]→place	0.01	0.77
10	[geolocation]→viewer	0.01	0.76

4) COLLABORATION PATTERN DISCOVERING FOR WEB APIS

As we have mentioned before, the rules in which tags belong to the same Web API will be removed since they may be improper to express the association between APIs, and the remainder strong API association rules can be adopted as API collaboration patterns. Table 6 shows the top 15 Web API collaboration patterns discovered from tag association rules. In the following, we take the 3rd and 10th collaboration patterns in Table 6 as examples to explain their meanings.

TABLE 6. Top 5 web API tag association rules (including 2,3 and 4-itemsets).

ID	Web API Tag Frequent Itemsets	Support	Confidence
1	[usa]→tts	0.02	0.77
2	[weather]→display	0.02	0.74
3	[weather]→viewer	0.02	0.74
4	[weather]→place	0.02	0.72
5	[global]→webhook	0.02	0.72
6	[display, video]→map	0.05	1
7	[viewer, video]→map	0.04	1
8	[viewer, video]→display	0.04	1
9	[viewer, search]→map	0.04	1
10	[viewer, search]→display	0.04	1
11	[display, viewer, video]→map	0.04	1
12	[display, viewer, search]→map	0.04	1
13	[display, place, video]→map	0.04	1
14	[viewer, place, video]→map	0.04	1
15	[viewer, place, video]→display	0.04	1

The pattern ''[weather]→viewer'' indicates that the Web APIs with ''weather forecast'' functionality and the Web APIs with ''viewer'' functionality are frequently combined to create applications. This indicates that the combination of functionalities ''weather forecast'' and ''viewer'' is very popular in mashup creation. In a similar manner, the pattern ''[viewer, search]→display'' indicates that the combination

of functionalities ''viewer'', ''search'' and ''display'' is popular in mashup creation.

Being aware of Web API collaboration patterns can help developers refine their requirements and build mashup applications more efficiently. For example, when a developer wants to develop a weather-forecast related mashup application, according to the mined API collaboration patterns, she/he can easily find out what combinations of functionalities are popular in mashup creation. For example, she/he may find that the ''[weather]→viewer'' API collaboration pattern is interesting and satisfies her/his requirement. Hence, she/he realizes that besides the weather forecast function, it's better to incorporate another functionality ''viewer'' into her/his mashup application. Finally, she/he search the Web API repository for the most appropriate Web APIs with tags ''weather'' and ''viewer'', and combine them to create the mashup application.

C. EVALUATION

In order to evaluate the proposed WACP approach, we compare it with the following two baseline approaches.

- **Baseline1:** WACP without tag expansion (WACP*). This approach is a reduced version of WACP, and is used to find how much effect the tag expansion has on the performance of WACP.
- **Baseline2**. This baseline, which was proposed in [16], used an Apriori-based approach to mine 2-itemset tag association rules between Web APIs. However, it did not take inconsistencies, typos and proliferation of synonyms with tags into consideration.

For the convenience of comparison, we use only the 2-itemset tag association rules with confidence above 0.5 in the experiments for evaluation. We use three evaluation metrics to evaluate the performance of Web API collaboration pattern mining: precision, recall, and F-measure, which are defined as follows:

$$Precision = \frac{F_{true}}{F_{true} + F_{false}} \tag{6}$$

$$Recall = \frac{F_{true}}{F_{true} + UF_{true}} \tag{7}$$

$$F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{8}$$

where F_{true} is the number of realistic ones of the mined Web API collaboration patterns, F_{false} is the number of unrealistic ones of the mined Web API collaboration patterns, and UF_{true} is the number of undiscovered but realistic Web API collaboration patterns. We use the tag association rules mined from the tags of mashups as benchmarks to test whether a tag association rule mined from the tags of Web APIs is a realistic API collaboration pattern or not. This is because that a mashup application is usually consisted of two or more Web APIs, and thus its tags can reflect the combination of different functionalities of Web APIs. If a tag association rule mined from Web APIs belonging to the mashups' tag association rule, it can be viewed as a realistic API collaboration pattern, otherwise it is an unrealistic pattern.

FIGURE 3, FIGURE 4 and FIGURE 5 show the precision, recall and F-measure performance of WACP and the baseline approaches. As we can see, the precision of the WACP is significantly higher than the other two approaches. The recall of the three approaches are very similar, and it is significantly affected by the support threshold. The F-measure, which is an aggregate of precision and recall, also indicates that WACP outperforms the other two approaches significantly. The baseline approach WACP* (WACP without tag expansion) slightly outperforms the other one proposed in [16], indicating that tag cleaning and synonyms consolidation do have positive effects on tag association mining. And the observation that WACP significantly outperforms WACP* indicates that, tag expansion is a crucial step to improve tag association mining and collaboration pattern discovering for Web APIs.

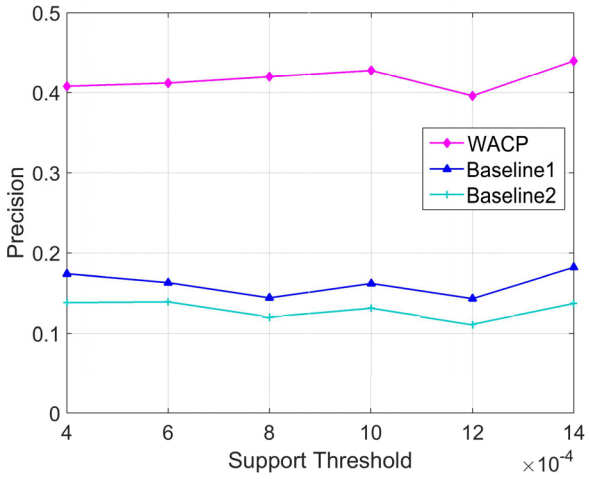


FIGURE 3. Precision comparison.

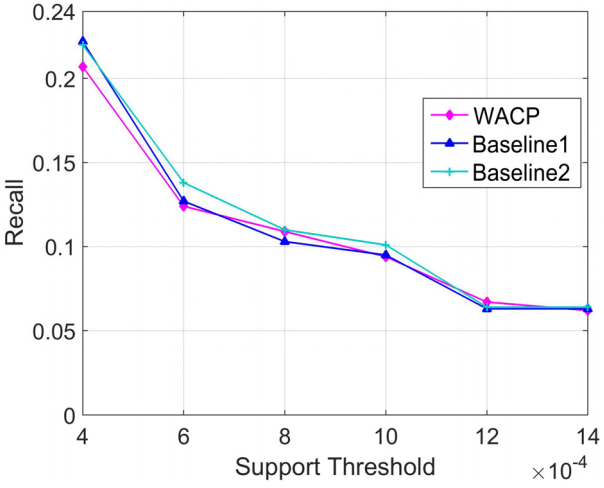


FIGURE 4. Recall comparison.

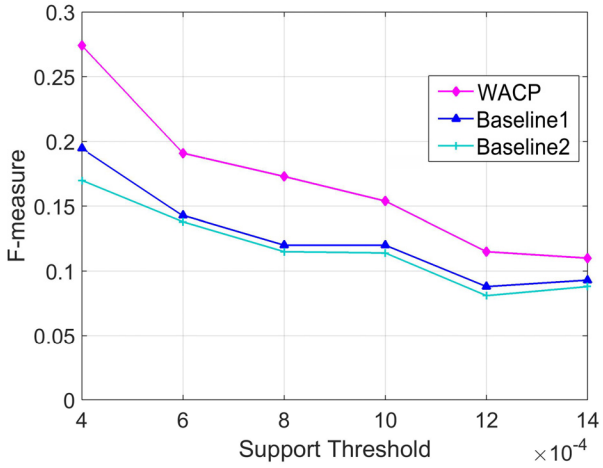


FIGURE 5. F-measure comparison.

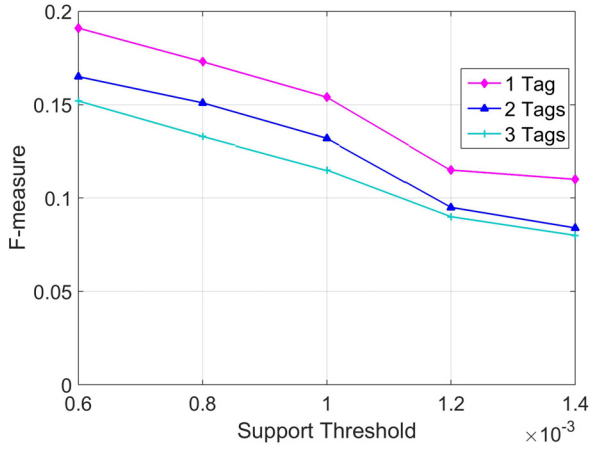


FIGURE 6. The influence of the number of API tags expanded on F-measure.

Furthermore, we also conducted an experiment to evaluate how the number of tags expanded for each Web API influences the performance of API collaboration pattern mining. FIGURE 6 shows the results of this experiment. As we can observe, the F-measure performance of one tag expansion is higher than that of two tag expansion and three tag expansion. This observation indicates that the proposed Web API collaboration pattern mining approach has the best performance when expanding one tag for each Web API. When more tags are extracted from the description text of Web APIs, according to TF-IDF, the less important tags (keywords) may cause noises to tag association rule mining and thus degrade the performance.

V. CONCLUSION

Based on the Web API collaboration history in mashup creation, this paper proposed an approach to mining the association rules between API tags which are used for annotating the functionalities of Web APIs. A strong association between API tags indicates that the corresponding tags have frequently co-occurred in API combinations for mashup creation and thus their association can be deemed as a typical API collaboration pattern. To overcome the data sparsity of API tags, the TF-IDF method was used to extract the keywords in the Web API description documents as new tags to enrich the tag sets of Web APIs. And to improve the quality of API tags, word lemmatization and synonyms consolidation were performed. Finally, a FP-growth-based algorithm was developed to mine the strong association rules among Web API tags. A set of experiments was conducted on a real-world Web API dataset and the experimental results validated the proposed approach.

In the future work, we plan to apply the mined API collaboration patterns to Web API recommendation for efficient mashup creation in Web of Things.

REFERENCES

[1] D. Guinard and V. Trifa, "Towards the Web of things: Web mashups for embedded devices," in *Proc. Int. World Wide Web Conf. (WWW), Workshop Mashups, Enterprise Mashups Lightweight Composition Web (MEM)*, Madrid, Spain, Apr. 2009, pp. 1506–1518

[2] D. Zeng, S. Guo, and Z. Cheng, "The Web of things: A survey," *J. Commun.*, vol. 6, no. 6, pp. 424–438, 2011.

[3] D. Guinard, V. Trifa, F. Mattern, and E. Wilde, "From the Internet of Things to the Web of things: Resource-oriented architecture and best practices," in *Architecting the Internet of Things*, New York, NY, USA: Springer, 2011, pp. 97–129.

[4] D. Guinard, V. Trifa, and E. Wilde, "A resource oriented architecture for the Web of things," in *Proc. Int. Things (IOT)*, Tokyo, Japan, Nov./Dec. 2010, pp. 1–8.

[5] D. Guinard, "A Web of things application architecture—Integrating the real-world into the Web," Ph.D. dissertation, Inst. Pervasive Comput., ETH Zurich, Zurich, Switzerland, 2011.

[6] D. Guinard, "Mashing up your Web-enabled home," in *Proc. Int. Conf. Web Eng. (ICWE)*, Vienna, Austria, Jul. 2010, pp. 442–446.

[7] M. Blackstock and R. Lea, "IoT mashups with the WoTKit," in *Proc. 3rd Int. Things (IOT)*, Wuxi, China, Oct. 2012, pp. 159–166.

[8] F. Leymann, D. Roller, and M. T. Schmidt, "Web services and business process management," *IBM Syst. J.*, vol. 41, no. 2, pp. 198–211, 2002.

[9] J. Zhang, W. Tan, J. Alexander, I. T. Foster, and R. K. Madduri, "Recommend-as-you-go: A novel approach supporting services-oriented scientific workflow reuse," in *Proc. IEEE Int. Conf. Services Comput. (SCC)*, Washington, DC, USA, Jul. 2011, pp. 48–55.

[10] Z. Zhou, Z. Cheng, L.-J. Zhang, W. Gaaloul, and K. Ning, "Scientific workflow clustering and recommendation leveraging layer hierarchical analysis," *IEEE Trans. Services Comput.*, vol. 11, no. 1, pp. 169–183, Jan./Feb. 2018.

[11] A. L. Edgardo and J. A. García-Macías, "TinySOA: A service-oriented architecture for wireless sensor networks," *Service Oriented Comput. Appl.*, vol. 3, no. 2, pp. 99–108, 2009.

[12] Z. Zhou, D. Zhao, L. Liu, and P. C. K. Hung, "Energy-aware composition for wireless sensor networks as a service," *Future Gener. Comput. Syst.*, vol. 80, pp. 299–310, Mar. 2018.

[13] L. Yao, X. Wang, Q. Sheng, W. Ruan, and W. Zhang, "Service recommendation for mashup composition with implicit correlation regularization," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, New York, NY, USA, Jun./Jul. 2015, pp. 217–224.

[14] W. Xu, J. Cao, L. Hu, J. Wang, and M. Li, "A social-aware service recommendation approach for mashup creation," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Santa Clara, CA, USA, Jun./Jul. 2013, pp. 107–114.

[15] B. Xia, Y. Fan, W. Tan, K. Huang, J. Zhang, and C. Wu, "Category-aware API clustering and distributed recommendation for automatic mashup creation," *IEEE Trans. Services Comput.*, vol. 8, no. 5, pp. 674–687, Sep. 2015.

[16] K. Goarany, G. Kulczycki, and M. B. Blake, "Mining social tags to predict mashup patterns," in *Proc. 2nd SMUC*, Toronto, ON, Canada, Jan. 2010, pp. 71–78.

[17] Y. Ni, Y. Fan, K. Huang, J. Bi, and W. Tan, "Negative-connection-aware tag-based association mining and service recommendation," in *Proc. 12th Int. Conf. Service Oriented Comput.*, Paris, France, Nov. 2014, pp. 419–428.
[18] Z. Gao *et al.*, "SeCo-LDA: Mining service co-occurrence topics for recommendation," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, San Francisco, CA, USA, Jun./Jul. 2016, pp. 25–32.
[19] Z. Gao, Y. Fan, C. Wu, W. Tan, and J. Zhang, "Service recommendation from the evolution of composition patterns," in *Proc. IEEE Int. Conf. Services Comput. (SCC)*, Honolulu, HI, USA, Jun. 2017, pp. 108–115.
[20] Y. Han, S. Chen, and Z. Feng, "Mining integration patterns of programmable ecosystem with social tags," *J. Grid Comput.*, vol. 12, no. 2, pp. 265–283, 2014.
[21] M. Tang, F. Xie, B. Cao, S. Lyu, and J. Liu, "Exploring Web services from a network perspective using multi-level views," *J. Web Eng.*, vol. 15, nos. 5–6, pp. 501–520, 2016.
[22] C. Y. Shi, C. J. Xu, and X. J. Yang, "Study of TF-IDF algorithm," *J. Comput. Appl.*, vol. 29, no. B06, pp. 167–170, 2009.
[23] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proc. Int. Conf. 20th VLDB*, Santiago, MN, USA, Sep. 1994, pp. 487–499.
[24] J. W. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Dallas, TX, USA, May 2000, pp. 1–12.

MINGDONG TANG received the B.S. degree in electrical engineering from Tianjin University, Tianjin, China, in 2000, the M.S. degree in control engineering from Shanghai University, Shanghai, China, in 2003, and the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2010.

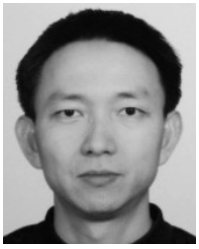
He is currently a Professor with the School of Information Science and Technology, Guangdong University of Foreign Studies, Guangzhou, China. He has published more than 100 peer-reviewed scientific research papers in various journals and conferences. His research interests include service-oriented computing, software engineering, and data mining.

Dr. Tang is a member of China Computer Federation, IEEE, and ACM.

YANMIN XIA received the B.S. degree in software engineering from Jishou University, Zhangjiajie, China, in 2016.

She is currently pursuing the M.S. degree in computer science with the Hunan University of Science and Technology, Xiangtan, China. Her research interests include service-oriented computing and data mining.

Ms. Xia is a member of China Computer Federation.



BING TANG received the B.S. and Ph.D. degrees in computer science from the Wuhan University of Technology, Wuhan, China, in 2005 and 2010, respectively.

He is currently an Associate Professor with the School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan, China. His research interests include software engineering and distributed systems.

Dr. Tang is a member of China Computer Federation, IEEE, and ACM.

YONGMEI ZHOU received the M.S. degree in computing science from Central South University, Changsha, China, in 2006.

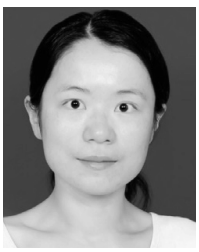
She is currently a Professor with the School of Information Science and Technology, Guangdong University of Foreign Studies, Guangzhou, China. Her research interests include natural language processing, software engineering, and data mining.

Prof. Zhou is a member of China Computer Federation.

BUQING CAO received the Ph.D. degree in computer science from Wuhan University, Wuhan, China, in 2011.

He is currently an Associate Professor with the School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan, China. His research interests include software engineering and data mining.

Dr. Cao is a member of China Computer Federation, IEEE, and ACM.



RONG HU received the Ph.D. degree in computer science from the Nanjing University of Technology, Nanjing, China, in 2015.

She is currently an Associate Professor with the School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan, China. Her research interests include software engineering and service-oriented computing.

Dr. Hu is a member of China Computer Federation.