Mining Social Tags to Predict Mashup Patterns

Khaled Goarany
Virginia Polytechnic Institute and State University
Falls Church, VA, USA
goarany@vt.edu

Gregory Kulczycki
Virginia Polytechnic Institute and State University
Falls Church, VA, USA
gregwk@vt.edu

M. Brian Blake
University of Notre Dame
Notre Dame, IN, USA
m.brian.blake@nd.edu

ABSTRACT
In the past few years, tagging has gained large momentum as a user-driven approach for categorizing and indexing content on the Web. Mashups have recently joined the list of Web resources targeted for social tagging. In the context of the social Web, a mashup is a lightweight technique for integrating applications and data over the Web. Crafting new mashups is largely a subjective process motivated by the users' initial inspiration. In this paper, we propose a tag-based approach for predicting mashup patterns, thus deriving inspiration for potential new mashups from the community's consensus. Our approach applies association rule mining techniques to discover relationships between APIs and mashups based on their annotated tags. We also advocate the importance of the mined relationships as a valuable source for recommending mashup candidates while mitigating for common problems in recommender systems. We evaluate our methodology through experimentation using real-life dataset. Our results show that our approach achieves high prediction accuracy and outperforms a direct string matching approach that lacks the mining information.

Categories and Subject Descriptors
H.2.8 [Database Management]: Database Applications – Data Mining; H.3.5 [Information Storage and Retrieval]: Online Information Systems – Web-based services; H.5.3 [Information Interfaces and Presentation]: Group and Organization Interfaces – Collaborative computing. Evaluation/methodology. Web-based interaction.

General Terms
Experimentation. Performance.

Keywords
Social tags, Mashup, Web mining, User-generated content.

1. INTRODUCTION
In the past few years, tagging has gained large momentum as a user-driven approach for categorizing and indexing resources on the Web. Thousands of photos on Flickr [10], videos on YouTube [32], and bookmarks on Delicious [4] are being tagged daily. The motivations for tagging range from content organization and retrieval to social practices [35]. Recent research have studied the use of tags in recommending Web resources [25][16][5][34]. In this line of research, tags are viewed as a rich, yet cheap, source of collective intelligence representing the community consensus on what concept(s) best describe a Web resource.

Recently, mashups have joined the list of Web resources targeted for social tagging. In the context of the social Web, a mashup has evolved as a user-centric lightweight approach for integrating applications and data over the Web. The fast adoption of mashups by online communities is motivated by its promise of enabling nontechnical users to integrate existing applications in new and innovative ways. This promise is further enhanced considering the growing number of user-friendly mashup tools, such as Yahoo Pipes [31], Intel Mash Maker [14] and IBM Mashup Center [13]. Furthermore, developers see mashups as a new approach for marketing their applications to a wider consumer base.

ProgrammableWeb [20] is a popular online community built around user-generated mashups, where users collaborate in posting, tagging and rating Web APIs and mashups. This social aspect has attracted the attention of a number of researchers, and enabled the study of many interesting phenomena related to online user behavior and mashup creation trends [27][33][29]. In our work, we utilize the ProgrammableWeb’s repository, to discover relationships between APIs and mashups based on their annotated tags, and use these relationships to predict interesting mashup patterns. Relationships are discovered using association rule mining techniques. We evaluate our methodology through experimentation on real-life dataset crawled from ProgrammableWeb. Our results demonstrate that our approach achieves high pattern prediction accuracy with significant performance gains with respect to a state-of-the-art direct string matching approach. Our work is distinguished by three key contributions:

1. A mining approach for discovering mashup patterns from user-generated tags - From a skeptic point of view, tags are often considered as a form of uncontrolled vocabulary that suffers from inconsistencies, typos, proliferation of synonyms and unstructured organization. By using empirical evaluation, we show that social tags can be used effectively to detect patterns within online communities despite these characteristics. In our experimentation, we use tags without any prior cleaning, stemming or consolidation of terms. Our results show that accurate patterns can be consistently discovered from raw tags by applying simple mining techniques that only consider tag pair relationships.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Proceedings of the 2nd International Workshop on Search and Mining User-generated Contents (SMUC 2010), October 30, 2010, Toronto, Canada. Copyright 2010 ACM.
A mashup recommendation methodology that mitigates for the cold start problem - The mined tag relationships constitute a valuable source of mashup candidate recommendations for APIs that have no prior usage or community rating information. We hence provide a solution for the cold start problem [24] typically experienced by recommender systems, where no recommendations can be generated for newly added items due to their lack of rating and usage history.

A new heuristic for mashup recommendations to balance the long tail effect - Another characteristic of online communities is the emergence of the long tail effect [18], where few items are constantly reused by the community leading to continuous increase in their popularity. The majority of items however are marginalized because of their limited usage and hence grow even less popular. This holds true for the ProgrammableWeb community, as stated in [27] and confirmed by our own analysis. By resorting to tags, we can create a ripple effect in this community, where less popular APIs are recommended based on their tag similarity to the more popular APIs. This can benefit both API developers by marketing their less known APIs, as well as consumers by exposing them to potentially interesting APIs that they didn’t know of before.

While we apply our approach to APIs and mashups, the use of tags in recommendations is a general concept applicable to other types of Web resources, where all the aforementioned benefits are of equal importance.

The remaining sections are organized as follows. Section 2 is a survey of related work. In Section 3, we present our solution methodology, including a detailed description of our mining approach and its potential applications. The results and analysis of our experimentation are presented in Section 4. We conclude the paper and outline our plans for future work in Section 5.

2. RELATED WORK
ProgrammableWeb has attracted the attention of many researchers as a popular online repository of mashups. In [27], the authors use ProgrammableWeb to study the API and tag usage patterns. They employ clustering techniques to discover relationships between user categories and the usage of both the API and tags. Their work concludes that users with geographical proximity tend to use the same APIs and tags and favor the same mashups. By studying the entropy of mashups created in ProgrammableWeb, they also confirm that, generally, users tend to reuse popular mashup patterns. The authors of [33] perform surveys to discover expectations of non-programmers from mashups, and the level of expertise needed to generate them. Their work leverages the reasoning that mashups constitute a valuable tool for empowering non-developers on the Web. One of their findings is that the mashup usefulness perceived by the end users is more important than the expected difficulty of creating it. Their results are particularly useful for designers of Web mashup tools. In [29], the authors categorize mashups surveyed from ProgrammableWeb under the following categories, which we refer to later in this paper:

- **Aggregation**: mashups aggregating data through multiple APIs.
- **Alternate UI**: mashups offering alternative interfaces for interacting with data from one or more APIs.
- **Personalization**: mashups extracting user-specific data through APIs.
- **Focused View**: mashups returning a subset of data from an API based on a query.
- **Real-Time Monitoring**: mashups monitoring real-time updates in websites through their APIs.

Research in mashup recommendations is rooted in the automatic discovery of Web service compositions. Work in this field mostly follows a semantic-based approach that relies on knowledge representation and ontology-based techniques. A survey can be found in [22]. These techniques aim at representing services’ functionalities and attributes using formal ontologies. Compositions are then discovered by applying systematic reasoning and inference methodologies on the ontological concepts. The Resource Definition Framework (RDF) [23] and Web Ontology Language (OWL) [17] are two standards used in semantically annotating services. While this approach offers a formal and precise way of describing a service offering, building ontologies and annotating services with formal concepts remains largely a manual process, guided by domain experts, and hence complex and time consuming. This constitutes a bottleneck in practical application of semantic techniques, hindering their widespread adoption by service providers. Moreover, it has been argued that the insufficient involvement of users in the construction of domain ontologies has rendered them non-intuitive from the users’ perspective [12]. Work in [2], [21], and [7] propose alternatives to mitigate for the semantic techniques shortcomings, by relying on syntactical-based matching of services. For example, in [2] an approach is proposed for matching services based on the analysis of human naming tendencies within Web Service Description Language (WSDL) specifications [28]. The approach studies the dependency between the message names of services and their mashability. We use a similar approach to [2] as a baseline in our evaluation.

In recent years, mashups have evolved as a light-weight user-centric form of compositions created by combining Web APIs. In the Web context, an API can refer to RESTful services [19], RSS feeds, Javascript widgets, or WSDL-based Web services, among other forms of Web accessible functionalities. The user-centric nature of mashups has encouraged its adoption by online communities such as ProgrammableWeb, where users collaborate in posting and tagging both APIs and mashups.

In our work, we consider social tags as a hybrid solution between the semantic and syntactical approaches used with Web service compositions. First, tags describe the functionalities of APIs and mashups from a user perspective, and hence their descriptions stem from practical usage. Also, the tags are also collectively provided by the community, which disseminates the creation effort. Moreover, tags are a generic form of descriptors for Web resources, and hence can be used to annotate any Web API regardless of its underlying protocol or standards. While tags lack the consistency and rigor of a formal ontology, we demonstrate through our experimentation that they are still of great benefit for discovering relationships between different APIs and for predicting interesting mashups patterns.

Among the research employing social tags for Web services discovery is [9], where a tag-based clustering technique is proposed for establishing similarity between services based on similarity between their tag clouds. Their work considers the tag
cloud as a lightweight form of ontology that can be used to index services and consequently retrieve services that match a user query. A similar approach is proposed in [3], where tags are used to discover and compose services through AI techniques. In this research, the rational is that users are more likely to use vocabulary from the tag cloud than from a formal ontology. We share the same rational, but we employ an empirical evaluation using real-life data to show the usefulness of mining tags in discovering mashup candidates.

Tag-based recommendation has also been explored in [25], which evaluates different algorithms for predicting users' preferences for movies based on their preferences for the movies’ associated tags. This work builds on research in [16], which proposes a cluster-based algorithm for recommending web-pages based on the pages users have tagged, and [5], which creates tag-based user profiles to use in recommendations. In [34], a collaborative filtering approach is proposed that discover similarity between users based on the semantic distance among their tags. In our work, we apply association rule mining to discover relationships between tags, and use these relationships in predicting mashup patterns. Recommending mashup candidates and API usage scenarios is among the potential applications of our approach.

Mashup recommendation is another area of research relevant to this work. A mashup recommender tool, called MashupAdvisor, is proposed in [8], which provides design-time assistance to mashup creators. To generate recommendations, the MashupAdvisor keeps a repository of mashups and uses it to calculate co-occurrence patterns between each pair of concepts. Concepts are extracted from APIs inputs and outputs. Similarity between concepts is established based on domain independent thesaurus and domain dependent ontologies. Compared to MashupAdvisor, our proposed approach relies on user-assigned tags, instead of API inputs and outputs, in generating recommendations. We believe that tags represent a more comprehensive set of concepts describing the API functionalities and hence our recommendations would potentially enable the discovery of a wider range of interesting APIs. Our experiments show that we can generate recommendations without the need of constructing a domain ontology, which is generally perceived as an impractical requirement. Intel Mash Maker [14] is another tool that enables users to customize the content of browsed websites by adding functionalities, visualizations or data from other websites. The recommendations provided by the tool are based on the content of the Web resource, in their case the currently browsed website, rather than on mining the tags annotating the resource as in our case.

3. TECHNICAL APPROACH

Our proposed solution is based on the hypothesis that mining social tags can be used as a powerful tool for predicting mashup patterns. In this section we describe in details our methodology in mining tags, and predicting mashup patterns. It also provides an overview of the applications and benefits we envision for our approach.

3.1 Mining Mashup Tags

The association rule mining problem is formally defined in [1] as follows: Let $I = \{i_1, i_2, ..., i_n\}$ be a set of items. An itemset $X$ with $k$ items from $I$ is called a $k$-itemset. Let $DB$ be a set of transactions, where each transaction $A$ is an itemset such that $A \subseteq I$. Given an itemset $X \subseteq I$, a transaction $A$ contains $X$ if and only if $X \subseteq A$. An association rule is an implication of the form $X \Rightarrow Y$ where $X \subseteq I$, $Y \subseteq I$ and $X \cap Y = \phi$. The rule $X \Rightarrow Y$ has support $s$ if $s\%$ of transactions in $DB$ contain $X \cup Y$. The association rule holds in the set of transactions $DB$ with confidence $c$ if $c\%$ of transactions in $DB$ that contain $X$ also contains $Y$.

The problem of mining association rules is to find all rules whose support and confidence are higher than a given minimum support and confidence thresholds. For an association rule $X \Rightarrow Y$, we can calculate:

$$\text{support}(X \Rightarrow Y) = \frac{\text{support}(X \cup Y)}{\text{support}(X)}$$ and $$\text{confidence}(X \Rightarrow Y) = \frac{\text{support}(X \cup Y)}{\text{support}(X)}$$

In our work, we model ProgrammableWeb as a set of mashups $M$, and a set of tags $T$. Each mashup in $M$ is comprised of one or more tags from $T$. These tags either annotate the mashup, or one of its Web APIs. In this context, mashups correspond to the transactions, and tags correspond to the items.

By applying association rule mining, we aim to discover tag association rules on the form $t_i \Rightarrow t_j$, where $\{t_i, t_j\} \subseteq T$. To achieve this goal, we apply the following steps:

1. For each mashup, we generate all possible tag pairs $\{t_i, t_j\}$, where:
   a. $t_i$ and $t_j$ are two tags annotated to two different APIs used by the mashup, or
   b. $t_i$ is annotated to one of the mashup APIs, and $t_j$ is annotated to the mashup itself.
2. We construct a knowledge base of all generated tag pairs. Each pair is assigned a count, representing the number of mashups where the pair is encountered.
3. These counts are used to calculate support and confidence metrics for each pair.

For example, consider the “Local Twitter Trends” mashup, listed on ProgrammableWeb. Local Twitter Trends displays the most posted terms on Twitter per location, by integrating data from Twitter [26] and Google Maps [11] APIs. In ProgrammableWeb, Google Maps is annotated with the tag “mapping”, and twitter is annotated with “microblogging”, and “messaging”. Local Twitter Trends is annotated with “microblogging”, “mapping”, “local”, and “trends”. By following steps 1.a and 1.b described earlier, we generate the following set of all possible tag pairs for this mashup:

$$\{\{\text{mapping, microblogging}\}, \{\text{mapping, messaging}\}, \{\text{mapping, trends}\}, \{\text{mapping, local}\}, \{\text{microblogging, trends}\}, \{\text{microblogging, local}\}, \{\text{messaging, trends}\}, \{\text{messaging, local}\}$$

By repeating this process for each mashup crawled from ProgrammableWeb, we construct a knowledge base from all extracted tag pairs. This knowledge base enables us to calculate the support and confidence for each pair by tracking its repeatability. A frequent tag pair; a pair whose support and confidence are above specified thresholds, represents a discovered association rule.
3.2 Predicting Mashup Patterns

As can be seen in Section 3.1, the annotated tags essentially represent all the concepts related to the mashup. These concepts may describe the functionalities of the mashed up APIs (e.g. “mapping”, “microblogging”), the new functionalities introduced by the mashup (e.g. “trends”), or any other attributes the users find useful in describing the mashup (e.g. “local”).

In our model, each association rule discovered between these concepts is an indication of how frequently users integrate the concepts into mashups. We hence consider each rule as a mashup pattern. In the previous example, mapping \(\Rightarrow\) trends is considered a mashup pattern if support(mapping \(\Rightarrow\) trends) and confidence(mapping \(\Rightarrow\) trends) are above specified thresholds.

This mining approach can be significantly beneficial in a number of applications. First, we can use the discovered patterns in recommending mashup candidates within a set of APIs, based on their tags. For each two APIs in the set, we can generate a list of tag pairs from cross-multiplying all the tags annotating the two APIs. By consulting the knowledge base of association rules, we can assign a mashability score to the two APIs, based on the collective support and confidence values of their generated tag pairs. Accordingly, for each API in the set, we can recommend a list of APIs to use in mashups, ranked by their mashability score.

Moreover, since our mining takes into account mashup tags, our recommendations are inherently not limited to mashup candidates, but also include possible API usage scenarios. These scenarios capture potential customizations for the original functionalities provided by the APIs. For example, in 3.1 the mashup tag “trends” suggests a usage for the “microblogging” functionality of Twitter. This is particularly beneficial for the case of one-API mashups, i.e. mashups that only customize the functionality of a single API. In ProgrammableWeb, this category currently comprises more than half of the listed mashups.

We envision two major advantages in basing recommendations on tags instead of APIs. Typically, recommender systems experience a problem in generating recommendations for newly added items that have no usage history, also known as the cold start problem [24]. A new API in ProgrammableWeb is assigned a set of tags, both on creation and later by the community users. We are hence able, using our approach, to generate instant recommendations for new APIs based on their tags, if prior mined tag rules exist in the knowledge base. This is particularly useful due to the fact that, within the same community, users tend to reuse tags [27].

The other advantage of tag-based recommendations is the potential to reduce the long tail phenomenon detected in ProgrammableWeb. Figure 1 depicts the distribution of a randomly selected subset of APIs from ProgrammableWeb and their corresponding mashup counts. As can be seen in the figure, the majority of the APIs contribute to 3 or fewer mashups each, while very few APIs in the community, those considered as the most popular, are continuously reused in most of the listed mashups, despite the existence of other APIs offering similar functionality. With the presence of this trend in ProgrammableWeb, a mashup recommender system can easily fall into the cycle of continuously recommending the more popular APIs, based on their ever-growing usage, at the expense of the increasingly marginalized majority.

By using our mining approach, an API is picked up in recommendations based on the popularity of each of its tags. In our model, the popularity of a tag is a function of the popularity of all the APIs it annotates, and thus the less popular APIs benefit from their tag similarity to the more popular ones. As mentioned earlier, we envision this to be of significant value to both API developers and consumers.

4. EXPERIMENTATION AND EVALUATION

4.1 Dataset

In our experimentation, we employ a dataset consisting of 3647 mashups and 1574 tags, all crawled from ProgrammableWeb on January 5th, 2010. Out of the crawled mashups, 2047 are randomly selected to build a knowledge base, 800 are selected as a training set, and the remaining 800 are used as a test set. Based on the model described earlier, each mashup is related to one or more tags, which annotate the API(s) involved in the mashup, or annotate the mashup itself.

4.2 Experiments Details

Our baseline experiment (experiment 0), investigates the use of tags similarity in predicting mashup patterns, in the absence of historical information on tags mashability. In a similar approach to that described in [2], this experiment employs the Levenshtein algorithm [15] to measure similarity. The algorithm is used to calculate the edit distance between two tags \(t_1\) and \(t_2\), \(editDist(t_1, t_2)\), as the number of deletions, insertions, or substitutions required to transform \(t_1\) to \(t_2\). As an example, \(editDist(“blog”, “book”)\) is equal to 2.

Since the edit distance as described depends on the length of input tags, we normalize the results by dividing the distance by the length of the longest tag. As we are interested in measuring the similarity between the tags, rather than their difference, we use the following derived similarity metric:

\[
Sim(t_1, t_2) = 1 - \frac{editDist(t_1, t_2)}{Max(Length(t_1), Length(t_2))}
\]

In our second experiment (experiment 1), we evaluate the benefit of using the knowledge base, described in details in section 3, in predicting mashup patterns. The knowledge base contains a set of
9017 rules, representing 2-itemset tag association rules derived from 2047 mashups.

In both experiments, mashup predictions are evaluated using precision and recall metrics. Precision and recall are calculated as follows:

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

where $tp$ is number of true positives, $fp$ is number of false positives, and $fn$ is number of false negatives returned by the experiment. In experiment 0, precision and recall are measured at different tag similarity thresholds, while in the experiment 1 we vary association rules support and confidence thresholds.

### 4.3 Results and Analysis

The following graphs display the results from experiments 0 and 1.

**Figure 2. Precision and recall achieved by experiment 0 at different tag similarity thresholds**

Figure 2 illustrates the effect of varying the tag similarity threshold on the precision and recall of patterns prediction. As shown by the figure, the higher the similarity threshold, the higher the precision, but the lower the recall. Precision and Recall dew point, indicated by point (a) in the figure, is at 10%. The highest achieved precision is 47%, at a 100% similarity threshold, which occurs when both input tags are identical. Through further analysis of the results, it is found that the “Aggregation” mashup category, described in [29], dominates the returned predictions at this threshold, particularly for mashups aggregating data from APIs offering similar functionality. Examples include “Only Apple Stories”, a mashup aggregating Apple Inc. news queried from Digg [6] and Delicious APIs, both tagged with “bookmark”, and “Google vs Yahoo Maps”, a mashup displaying locations side-by-side from Google Maps and Yahoo Maps [30] APIs, both tagged with “mapping”.

The correlation results indicate that the tag similarity approach generally falls short in predicting mashup patterns. The Spearman correlation between tag similarity and tag mashup ratio; percentage of mashups where a tag occurs, is found to be 0.05. This indicates that tag similarity can be considered as a poor criterion for discovering patterns.

**Figure 3. Precision and recall achieved by experiment 1 at different support thresholds**

Figure 3 depicts the results obtained from experiment 1, where we predict mashup patterns based on a knowledge base of mined association rules. Precision and recall of patterns prediction are plotted against different tag pair support thresholds. As the support threshold increases, an increase in precision is observed, accompanied with a decrease in recall. The precision and recall dew point, indicated by point (b) in the figure, is at 55%; a 45% increase over the dew point of experiment 0.

The tag mining approach realizes a 0.8174 Spearman correlation between calculated support values for tag pairs and tag mashup ratio, indicating a powerful pattern prediction mechanism. Moreover, further analysis of the results proves that this approach has enabled us to predict tag mashability across all different mashup categories from [29].

As can be observed in Figure 3, the tag mining approach generally achieves higher precision values than those achieved by using tag similarity. This can be particularly useful for mashup recommendation systems that return a predetermined number of mashup candidates based on predicted tag mashability. In such a case, we would not be interested in recalling all possible recommendations, but rather interested in ensuring the highest precision possible for the returned results. Additionally, the calculated support values, used in predications, can also be used in ranking the list of returned recommendations.

Next, we study the effect of complementing the support threshold by a confidence threshold, calculated from the mined tag pairs. Figure 4 displays the precision and recall achieved for three exemplary confidence thresholds (0, 0.02, and 0.04) at a fixed support threshold of 0.001. As illustrated, increasing the confidence threshold leads to a slight increase in precision. However, this increase comes at the expense of a significant decrease in recall. This shows that confidence can be more beneficial as an additional restrictive criterion to filter out false predictions, in the cases where recall is of less importance. Again,
We envision a number of applications for our proposed accuracy in predicting patterns. Despite their often touted unfavorable characteristics, our results show that, while only considering 2-itemsets of raw tags that did not undergo any prior processing, we are able to achieve high accuracy in predicting patterns.

Our work proves through experimentation that social tags can be of significant benefit in mashup candidates, and API usage scenarios. Our methodology overcomes the cold start problem and the long tail phenomenon exhibited by other recommender systems that rely on historical usage information.

Our future work includes mining larger itemsets consisting of three or more tags, to discover more complex relationships. We are also planning to evaluate the use of different tag-based clustering techniques in predicting mashup categories.

5. CONCLUSION AND FUTURE WORK
In this paper, we demonstrate using empirical evaluation that we can apply association rule mining to social tags to learn and predict mashup patterns. Our work proves through experimentation that social tags can be of significant benefit despite their often touted unfavorable characteristics. Our results show that, while only considering 2-itemsets of raw tags that did not undergo any prior processing, we are able to achieve high accuracy in predicting patterns.

We envision a number of applications for our proposed methodology, including a tag-based recommendation system for mashup candidates, and API usage scenarios. Our methodology overcomes the cold start problem and the long tail phenomenon exhibited by other recommender systems that rely on historical usage information.

Our future work includes mining larger itemsets consisting of three or more tags, to discover more complex relationships. We are also planning to evaluate the use of different tag-based clustering techniques in predicting mashup categories.

6. REFERENCES


