A Fine-Grained API Link Prediction Approach Supporting Mashup Recommendation

[1] Qihao Bao, [1] Jia Zhang, [1]Xiaoyi Duan, [2]Rahul Ramachandran, [3]Tsengdar J. Lee, [1]Yankai Zhang, [1]Yuhao Xu,[4]Seungwon Lee, [4]Lei Pan, [2]Patrick Gatlin, [2]Manil Maskey
[1]Carnegie Mellon University -Silicon Valley, USA
[2]NASA/MSFC, USA
[3]Science Mission Directorate, NASA Headquarters, USA
[4]Jet Propulsion Laboratory, California Institute of Technology, USA
{qihao.bao, jia.zhang, xiaoyi.duan, yankai.zhang, yuhao@sv.cmu.edu, {rahul.ramachandran, patrick.gatlin, manil.maskey}@nasa.gov, tsengdar.j.lee@nasa.gov, {seungwon.lee, lei.pan}@jpl.nasa.gov

***Abstract*—Service (API) discovery and recommendation is key to the wide spread of service oriented architecture and service oriented software engineering. Service recommendation typically relies on service linkage prediction calculated by the semantic distances (or similarities) among services based on their collection of inherent attributes. Given a specific context (mashup goal), however, different attributes may contribute differently to a service linkage. In this paper, instead of training a model for all attributes as a whole, a novel approach is presented to simultaneously train separate models for individual attributes. Meanwhile, a latent attribute modeling method is developed to reveal context-aware attribute distribution. Experiments over real-world datasets have demonstrated that this fine-grained method yields higher link prediction accuracy.**

Keywords—Context-aware service recommendation; attribute model training; latent attribute distribution; mashup recommendation

I. Introduction

Computer-supported service (API) discovery and composition (mashup) has become a critical topic in the field of services computing. Earlier work on service discovery focuses on keyword-oriented text mining techniques to identify interested service candidates [1]. Semantic web techniques are applied to add fuzzy search and semantics-powered search ability [2]. Inspired by Amazon's recommendation scenarios as "people buy item A usually buy item B," Collaborative Filtering technique is applied in service discovery to recommend services used by peers with similar background and profiles [3].

In our previous work, we proposed to treat software services as social entities and build service social networks (SSN) [4]. By studying the social behaviors of services from their usage history (i.e., provenance), we developed algorithms to recommend and predict future service usages using social network analysis techniques. From then on, dozens of papers have appeared to report various approaches to study service usage provenance and rank service candidates under contextual conditions. For example, Zhong et al. [5] consider time series of service usages when recommending mashup components.

In recent years, researchers have started to leverage machine learning techniques to analyze service usage history. Zhong et al. [5] leverage the Latent Dirichlet Allocation (LDA) [6] technique from the information retrieval field to summarize topic distribution from API description documents. Gao et al. [7] apply the LDA technique to mine service co-occurrence relationships from past mashup history.

In this project, we aim to further study how to apply machine learning techniques, in a service social network, to incrementally train models from service provenance to better predict service linkages at runtime. Be more specific, given a mashup design purpose with some APIs already decided to be used in the mashup, how to recommend other APIs to co-exist in the mashup? In contrast to related work, we aim to study how fine-grained level attributes may contribute to context-aware API linkage prediction.

The motivation of this research comes from our Apache Incubator project - Climate Model Diagnostic Analyzer (CMDA) [8]. In the CMDA project, we originally constructed a CMDA REST service social network. Its comprising nodes are individual CMDA APIs that encapsulate inherent attributes. Using such a class-level model, the carried attributes may contribute to the connections among services; however, the relationship may be implicit. Fig. 1 shows an example of finding a subsequent API $S2$ following $S1$ in a mashup. Say that a user intends to design a mashup to calculate the zonal mean of an anomaly of a variable. This task can be achieved by a mashup that links the anomaly calculation API and the zonal mean calculation API. For the second step, there are two zonal-mean calculation API candidates with similar functions: one is for 2-dimensional variables; the other is for 3-dimensional variables. The attribute about the output data variable dimension from the first API (anomaly calculation) will be a determining factor to choose which zonal-mean calculation API to use in the second stage of the mashup, while other attributes do not contribute as such in this mashup-aware API recommendation problem. This example shows the necessity of

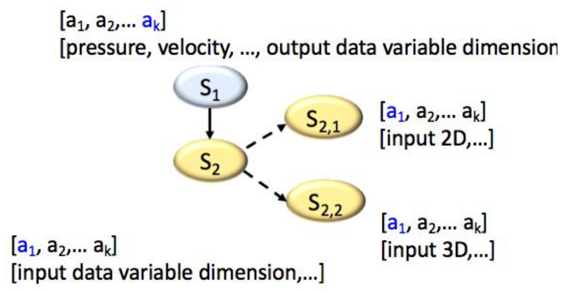


Fig. 1. Motivating example of the importance of attributes on service recommendation.

treating individual attributes as first-class citizen instead of hiding them under the surface of APIs.

Our contributions are summarized in three-fold. First is that we have developed a scalable attribute-level data model, featuring scalability and extensibility. We have extended Multiplicative Attribute Graph (MAG) [9] model to represent node profiles featuring rich categorical attributes, while relaxing its constraint of requiring *a priori* knowledge of predefined attributes. LDA is leveraged to dynamically identify attributes based on attribute modeling, and multiple Gaussian fit is applied to find global optimal values.

The second contribution is that we have seamlessly integrated the latent relationships between API attributes as well as observed network structure based on historical API usage data. Such a layered information model enables us to predict the probability of a link between two APIs based on their attribute link affinities carrying a variety of information including meta data, semantic data, historical usage data, as well as crowdsourcing user comments and annotations.

The third contribution is that we have developed a fine-grained context-aware mashup-API recommendation technique. On top of individual models trained for separate attributes, a dedicated layer is trained to represent the latent attribute distribution regarding mashup purpose, i.e., sensitivity of attributes to context. Thus, given the description of an intended mashup, the attributes sensitive to the goal will be identified, and corresponding attribute models will be exploited to compute the possibility of API linkages under the context. Such a layered model increases search accuracy.

The remainder of the paper is organized as follows. In Section II, we discuss related work. In Section III, we present our basic attibute-level model. In Sections IV and V, we explain the details of individual attribute model training and context-aware latent attribute distribution learning, respectively. In Section VI, we describe our service recommendation technique. In Section VII, we present experimental studies and analysis. In Section VIII, we draw conclusions.

II. Related Work

Link prediction remains a hot topic in social network study, aiming to infer interactions that would probably occur in the future given a snapshot of the current network. While all link prediction methods assigning a weight score for each pair of nodes, Liben-Nowell and Kleiberg summarize them into three categories [10]: methods based on node neighborhood, methods based on the ensemble of all paths and higher-level methods such as Unseen Bigrams, and methods based on clustering and low-rank approximation.

Most of research in the literature focus on examining similarity between nodes to provide an estimation whether they may be connected to each other. A number of algorithms are proposed to calculate profile similarity among nodes, leveraging information from various sources to form a vector of attributes for each node [11]. While each node is profiled as a vector of attributes, these attributes are typically used as features to classify nodes into clusters based on specific patterns. Such methods work well in many cases; however, similarity is just one way to study the connections between nodes [9].

To predict the link between two nodes, Kim and Leskovec [9] believe it is necessary to consider not only the similarity between them, but also the possibility that they connect to each other due to dissimilarity. Proposing the Multiplicative Attribute Graph (MAG) model [9], they estimate affinity matrices for node attributes. To simplify the problem, the original MAG assumes that each attribute is in binary format. Later researchers raise the bar a little and assume that every attribute of each node is drawn from a Bernoulli distribution [12]. This assumption still may not work in our science knowledge graph, however. Nodes in a service social network are usually profiled with crowdsourcing or information retrieval mechanism. Thus, their profiles contain a large amount of unstructured data in the form of text content, such as descriptions, and structured attributes that cannot be easily converted to a binary value to be estimated by Bernoulli distribution. Therefore, in our project we propose to further raise the bar and learn individual models for node attributes. Instead of using affinity matrix, we estimate affinity distribution covering continuous values.

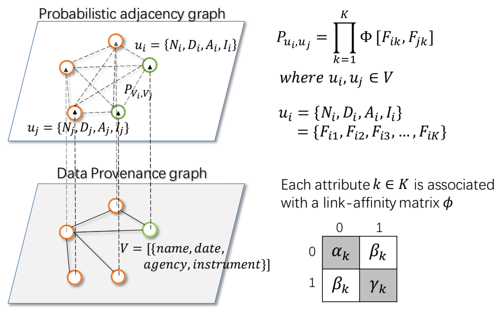
Some recent studies consider network structure could help profile nodes, as such information represents the property, history and behaviors of the nodes. A social-attribute network was developed which integrates network structure and node attributes by introducing attributes nodes into the graph [13]. Some researchers focus on extracting hidden attributes from nodes to create latent feature vectors [14]. In contrast to their work, we aim to study how individual attributes may contribute to context-aware API (node) connections in an SSN.

III. Information Model

A. Basic Attribute-Level Model

To study fine-grained API link prediction, we propose to link physical world and virtual world in a service social network, which we call Basic Attribute-Level Model (B-ALM). As shown in Fig. 2, B-ALM can be viewed as a two-layer graph: an overlay probabilistic adjacency graph is built on top of a provenance graph.

Definition 1: A basic attribute-level model (ALM) is a 5-tuple $M = (V, E, E', \{F_u\}, \{\Phi_k\})$, where V is a set of vertices and $\{F_u\}$ is a collection of categorical attribute vectors for each $u \in V$, $\{\Phi_k\}$ is a collection of link-affinity matrices over all K attributes ($k \in K$), each representing the probability of two vertices to link to each other based on the values of one particular attribute. E and E' are both sets of edges. E


Probabilistic adjacency graph
$u_i = \{N_i, D_i, A_i, I_i\}$
P_{V_i,V_j}
$u_j = \{N_j, D_j, A_j, I_j\}$
Data Provenance graph
$V = [\{name, date, agency, instrument\}]$
$P_{u_i,u_j} = \prod_{k=1}^{K} \Phi\,[F_{ik}, F_{jk}]$
$where\ u_i, u_j \in V$
$u_i = \{N_i, D_i, A_i, I_i\} = \{F_{i1}, F_{i2}, F_{i3}, \dots, F_{iK}\}$
Each attribute $k \in K$ is associated with a link-affinity matrix ϕ
0
1
0
α_k
β_k
1
β_k
γ_k
Fig. 2. Basic Attribute-Level Model overview.

comprises actual links each representing an existing linkage between a pair of vertices. E' comprises virtual links each representing the probability of a linkage between a pair of vertices.

Example: Consider a B-ALM comprising five nodes as shown in Fig. 2, where a node represents a data service carrying four attributes: name, date, agency, and instrument. Thus, node V_i has an attribute vector $F_i = \{N_i, D_i, A_i, I_i\}$ with four categorical attributes associated.

A link-affinity matrix Φ_n is used to bridge the gap between attribute values $\{F_u\}$ and the probability P_{u_i,u_j} of a link between a pair of nodes, say u_i, u_j. Each attribute F is associated with a link-affinity matrix ϕ.

B. Link Affinity

Link affinity can be categorized into four types: homophily, heterophily, core-periphery, and randomness [9]. Some node attributes may have positive affinities with other attributes. For example, each CMDA API possesses an attribute "sponsored project." APIs yielded from the same project have higher probability to be used together in a large-scale data analytics workflow. Thus, we can claim that attribute "sponsored project" has positive affinities (i.e., Homophily).

Some nodes attributes may also have negative affinities (Heterophily) with other attributes. For example, some APIs can plot a graph regarding rainforests temperature, whereas some APIs would benefit research on desert air humidity. These APIs may rarely be combined with each other; in other words, they do not have high probability to be linked together. We can claim that the attributes "focus on rainforest temperature?" and "focus on desert humidity?" have negative affinities (i.e., Heterophily). Core-periphery means values are relatively larger near a cohesion core and decrease as they locate further from the core. Randomness means values do not follow specific patterns.

A simple binary link-affinity matrix template is shown in Fig. 2 on the lower right part, where the attribute value is in binary. If both nodes bear the same value "0" or "1" for the *nth* attribute, their probability to link together will be α_n and γ_n, respectively.

In building a B-ALM network, for each attribute type $\boldsymbol{F_k}$, the corresponding link-affinity matrix $\boldsymbol{\Phi_k} \in \{\boldsymbol{\Phi}\}$ can be generated. With matrixes generated, the probabilistic values of edges between every pair of nodes can be calculated through:

$$P_{u_i,u_j} = \prod_{k=1}^{K} \Phi\left[F_{ik}, F_{jk}\right] \tag{1}$$

IV. Affinity Distribution Learning Using Semantic Data

MAG assumes that all attributes are predefined, and their values are discrete (either binary [9] in the earlier work or enumerable [14] in recent work). Since SSN is incrementally built, such binary or Bernoulli distribution assumptions will not reflect various conditions. Therefore, instead of using affinity matrix carrying discrete values, we decide to build affinity distribution carrying continuous values.

A. Model Definition

Equation (1) illustrates how to calculate link probability between every pair of nodes based on predefined affinity matrices provided by domain experts. It is yet to be considered how to estimate the affinity matrix if there is no prior knowledge that can be easily obtained and utilized to assign the value of affinity matrices. In addition, what if the attributes are not discrete, such as length, weight, and possibilities? Instead of presenting affinity relation with a 2×2 matrix shown in Fig. 2, we use a Bivariate Normal Distribution to capture such latent information for each attribute and use the Probability Density Function (PDF) value of learned distribution to indicate how relevant two nodes are, based on the attribute values of the two nodes and actual link between them. Rather than considering only True/False values for each attribute, we propose to use real number values instead. We thus revise Definition 1 as follows to reflect such a change.

Definition 1': An attribute-level model (ALM) is a 6-tuple $M = (V, E, A, P, \{F_u\}, \{\Theta_t\})$, where V is a set of vertices, and $\{F_u\}$ is a collection of vectors of |T| attributes for each $u \in V$, $\{\Theta_t\}$ is a collection of link-affinity distributions over T ($t \in T$), each representing the probability of two vertices to link to each other based on the value of one attribute of two nodes. E, P and A are all sets of edges. E comprises actual links each representing an existing linkage between a pair of vertices; P comprises virtual links each representing probability of a linkage between a pair of vertices; and A comprises virtual links predicted with link probability larger than a certain threshold.

While our ALM covers APIs as nodes in a service social network, it is intuitive to leverage their parameters and properties as attributes. However, things may become more complicated due to several significant reasons. First, although some APIs provide attributes, most of them are categorical and hence could not provide the quantifiable information of how each attribute contributes to an API. Second, the attributes provided by API metadata may not be accurate without considering description from service providers. Third, attribute extraction should be decided based on the targeted API sets for scalability and comparison efficiency. Fourth, manual labeling is labor intensive. In summary, when studying relationships among a collection of APIs, the actual properties used as attributes should be dynamically derived from the API description set.

B. Affinity Distribution Estimation

Therefore, we apply machine learning technique to predict network relationships inside of ALM, considering both API descriptions and historical API connections. Latent Dirichlet Allocation (LDA) [6] is applied to API description to extract API attributes and consider probability of each attribute as attribute value in ALM. Afterwards, attribute values and historical linkage information are used for learning link-affinity distribution. Our approach is illustrated in Fig. 3 using the plate notation, which is widely used to represent repeated variables by grouping them in a subgraph with a rectangle called 'plate' [15]. Each subgraph is duplicated many times with its comprising variables indexed by the repetition marked at the corner of a rectangle. Table I summarizes the notations used in our model.

The overall generative process of our API-based edge probability with network adjacency matrix can be described as follows:

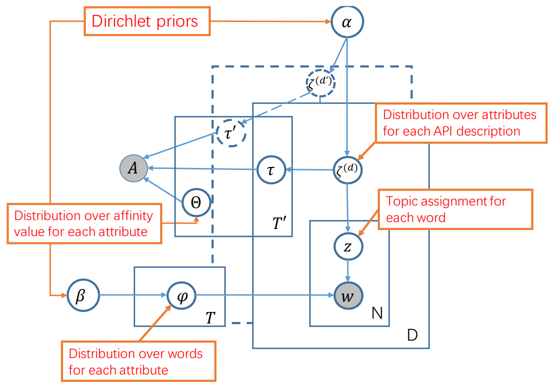

Dirichlet priors
Distribution over attributes for each API description
Topic assignment for each word
Distribution over affinity value for each attribute
Distribution over words for each attribute
α
β
$\zeta^{(d')}$
$\zeta^{(d)}$
τ'
τ
A
Θ
z
w
φ
T'
T
N
D

Fig. 3 Plate notation for generative process of edge probability in SSN.

```
1) For each attribute $t = 1:T$
        Draw $\varphi_t \sim Dirichlet(\beta)$
2) For each API description $d = 1:D$
        a) Draw $\zeta^{(d)} \sim Dirichlet(\alpha)$
        b) For each word token $i = 1:N$
                Draw an attribute $z_i^{(d)} \sim Multinomial(\zeta^{(d)})$
                Draw a word $w_i^{(d)} \sim Multinomial(\varphi_t)$
        c) $\mathcal{T}^{(d)} = \left(\tau_1^{(d)}, \tau_2^{(d)}, \ldots, \tau_{T'}^{(d)}\right) = \zeta^{(d)}$
3) For each attribute $t' = 1:T'$
        For each API description pair $(d, d') \in D \times D$
                Draw $\left(\tau_t^{(d)}, \tau_t^{(d')}\right) \sim Bivariate\ \mathcal{N}(\Theta)$
```

With the trained model, the probabilistic score indicating two APIs with description d and d' can be calculated as follows:

$$P_{d,d'} = \prod_{n=1}^{T'} \Phi_n \left[\tau_n^d, \tau_n^{d'}\right]$$

Given a threshold we can predict if two APIs may have a connection in the future:

$$A_{d,d'} = \begin{cases} 1, & if P_{d,d'} > threshold \\ 0, & Otherwise \end{cases}$$

1) Learning Distribution over API Attributes

Table 1. Notation used in affinity distribution estimation.

Symbol	Description
$\boldsymbol{D}$	Number of descriptions
$\boldsymbol{n}$	Number of words
$\boldsymbol{T}$	Number of attributes
$\boldsymbol{T'}$	Number of attributes
$\boldsymbol{\alpha}, \boldsymbol{\beta}$	The parameters of Dirichlet priors to the multinomial distribution $\zeta^{(d)}$ and $\varphi^{(Z)}$
$\boldsymbol{\varphi^{(Z)}}$	The parameters of multinomial distribution over words specific to description z
$\boldsymbol{\zeta^{(d)}}$	The parameters of multinomial distribution over attributes specific to description d
$\boldsymbol{\zeta^{(d')}}$	The parameters of multinomial distribution over attributes specific to another description d'
$\boldsymbol{z^{(n)}}$	The attribute assign to word n
$\boldsymbol{w}$	The word token in API description
$\boldsymbol{\tau}$	Probabilistic value of one attribute from $\zeta^{(d)}$
$\boldsymbol{\tau'}$	Probabilistic value of the same attribute as τ from $\zeta^{(d')}$
$\boldsymbol{\Theta}$	The parameters of Bivariate Normal Distribution over τ and τ'
$\boldsymbol{P}$	The probabilistic adjacency matrix
$\boldsymbol{A}$	The network generated given P with a series of coin flips

To tackle the issue of utilizing unstructured text data and transforming it to node attributes, we first apply a topic model following Dirichlet distribution. The affinity distribution is estimated based on the node attributes and known relationships between nodes.

A topic model is a type of statistical model for discovering abstract "topics" that occur in a collection of documents. Topic modeling is a frequently used text mining tool for discovering hidden semantic structures in a text body. Among others, Latent Dirichlet Allocation (LDA) [6] is a known generative statistical model allowing sets of textual observations to be explained by unobserved topics. In case of service discovery and recommendation, the relation between the attributes of services to service description is analogous to the topics to documents. We thus apply the LDA concept to mine attribute distribution over services.

As the first step, we present our idea as a probabilistic graphical model in Fig. 3. The parameters α and β are repository-level parameters, assumed to be sampled once in the process of generating a service repository. The variable ζ is a service-level variable, sampled once per service. Finally, the variables Z and W are word-level variables and are sampled once for each word in each service description.

As the second step, we integrate in past API connections. As shown in Fig. 3, for each pair of APIs whose attribute distributions are $\zeta^{(d)}$ and $\zeta^{(d')}$, respectively, let us consider how each of their attribute contributes to their actual connection (visible connection A). For every attribute, over a distribution of Θ, the affinity values (τ and τ') from the two APIs form a probability of their connection P.

In our model, we aim to learn the collection of latent parameters from provenance data. Expectation Maximization (EM) is applied to estimate the Generative Model. Our goal is to reason about the full joint distribution:

$$P(w, z, \varphi, \zeta | \alpha, \beta) = P(w, z | \varphi, \zeta) P(\varphi | \beta) P(\zeta | \alpha) \qquad (2)$$

We marginalize the model parameters out of the joint distribution, so that we can focus on the words in the repository (w) and their assigned attributes (z):

$$P(w, z | \alpha, \beta) = \int_{\Phi} \int_{\zeta} P(w, z | \Phi, \zeta) P(\varphi | \beta) P(\zeta | \alpha) d\zeta d\varphi$$

According to Bayes Rules we have:

$$P(w, z | \alpha, \beta) = P(w | \alpha, \beta) P(z | w, \alpha, \beta) \qquad (3)$$

For inference of the model, Gibbs sampling [16] is used to approximate the distribution of $P(z|w, \alpha, \beta)$, which is the attribute assignment given the word observation. By sampling each z_i, complete (or full) conditionals can be derived for each z_i in z:

$$P(z_i = j | z_{-i}, w; \alpha, \beta) = \frac{P(z_i, z_{-i}, w; \alpha, \beta)}{P(z_{-i}, w; \alpha, \beta)}$$

$$\propto \frac{n_{-i,j}^{(w_i)} + \beta,}{n_{-i,j}^{(\cdot)} + T\beta} \cdot \frac{n_{-i,j}^{(d_i)} + \alpha,}{n_{-\mathrm{i},(\cdot)}^{(d_i)} + D\alpha} \qquad (4)$$

where: d_i is the API description where word w_i occurs; z_i is the attribute assignment for word w_i; $n_{-i,j}^{(\cdot)}$ denotes a count of words under attribute j which does not include the current assignment z_i ; $n_{-i,j}^{(w_i)}$ is the number of times (ignoring position i) word w_i is assigned to attribute j; $n_{-i,j}^{(d_i)}$ is the number of times (ignoring position i) attribute j is used in API description d_i ; $n_{-i,(\cdot)}^{(d_i)}$ is the number of times (ignoring position i) all attributes are used in API description d_i. Intuitively, we can interpret the first ratio as the probability of word w_i under attribute j considering the number of words under attribute j; and the second ratio as the probability of attribute j in document d_i considering other attributes that are used in it.

At this stage, we only consider ALM when each attribute is conditionally independent from each other. Each attribute assigned to an API description would be given a probability, indicating the importance of the attribute over the API and there is no dependency between two attributes of one API description. Attributes with dependency will be studied in our future research.

However, traditional LDA is an unsupervised algorithm that models each document as a mixture of topics. We adopt the Labeled LDA (L-LDA) [6, 17], which is a supervised LDA by constraining the topic model to use only the topics that correspond to a document's (observed) label set. In our case, $\zeta^{(d)}$ is restricted to be defined only over the attributes that correspond to its labels Λ(d).

2) Learning Distribution over Affinity Values

We then assume each affinity distribution θ_l follows Bivariate normal distribution, $\theta_l \sim \mathcal{N}(\mu, \Sigma), l = 1,2,\dots,L$, where L is the number of attributes. A is the adjacency matrix of the graph:

$$A_{ij} = \begin{cases} 1, if\ node\ i\ and\ j\ have\ a\ link\ between\ them \\ 0, if\ node\ i\ and\ j\ have\ no\ link\ between\ them \end{cases}$$

Node attribute vector, $F = \{f_{ln}, n = 1,2,\dots,N, l = 1,2,\dots,L\}$, is obtained after applying the L-LDA method to the original node profiles in form of text content, which in our case is the API description. The likelihood given a graph can be represented as below:

$$P(A|\Theta) = \prod_{A_{ij}=1} P_{ij} \prod_{A_{ij}=0} (1 - P_{ij})$$
$$= \prod_{A_{ij}=1} \prod_{l=1}^{L} \theta_l[f_{il}, f_{jl}] \prod_{A_{ij}=0} (1 - \prod_{l=1}^{L} \theta_l[f_{il}, f_{jl}]) \quad (5)$$
$$where\ \theta_l \sim \mathcal{N}(\mu_l, \Sigma_l)$$

The calculation of θ_l can be presented as:

$$\theta_l[f_{il}, f_{jl}] = \frac{1}{\sqrt{4\pi^2\Sigma_l}} exp(-\frac{1}{2}\left((f_{il}, f_{jl}) - \mu_l\right)^T \Sigma_l^{-1} \left((f_{il}, f_{jl}) - \mu_l\right)) \quad (6)$$
$$where\ \mu_l = (E[f_{1l}], E[f_{2l}])$$
$$\Sigma_l = \left(Cov(f_{il}, f_{jl})\right), i = 1,2, j = 1,2$$

As part of the learning process, we can use maximum likelihood estimation to find affinity matrix distribution:

$$\arg\max P(A|\Theta)$$

Representing likelihood with $\mathcal{L}(\Theta) = \mathcal{L}(\mu, \Sigma)$, we aim to find $\mu = \{\mu_1, \mu_2, \dots, \mu_L\}\ and\ \Sigma = \{\Sigma_1, \Sigma_2, \dots, \Sigma_L\}$ to maximize:

$$\mathcal{L}(\Theta) = \mathcal{L}(\mu, \Sigma)$$
$$= Log(\prod_{A_{ij}=1} \prod_{l=1}^{L} \theta_l[f_{il}, f_{jl}] \prod_{A_{ij}=0} (1 - \prod_{l=1}^{L} \theta_l[f_{il}, f_{jl}])) \quad (7)$$

With such a likelihood function, we intend to find the parameters for the bivariate normal distributions of each attribute. It would maximize the edge probability if a pair of nodes can form a link between them, and minimize the edge probability if otherwise.

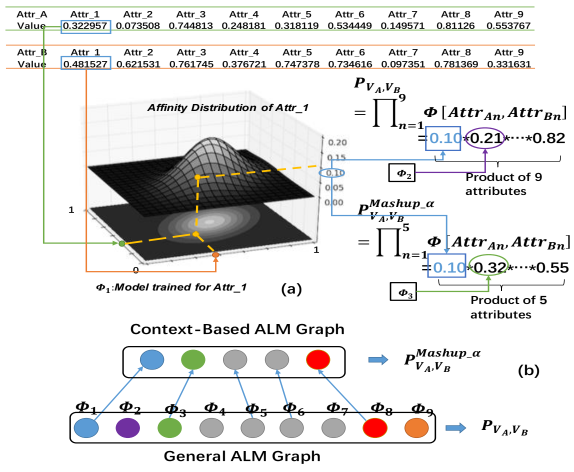
However, the above likelihood function could be hard to maximize when many attributes are related. Given any two nodes in an SSN, the possibility for them to be connected is rather small: $\#(A_{ij} = 1) \ll \#(A_{ij} = 0)$, which makes the adjacency matrix of the graph very sparse. To make the training process more effective, we consider only the attributes of adjacent nodes instead, so the likelihood equation can be simplified as follows:

$$\mathcal{L}(\Theta) = \mathcal{L}(\mu, \Sigma) = Log\left(\prod_{A_{ij}=1} \prod_{l=1}^{L} \theta_l[f_{il}, f_{jl}]\right) \quad (8)$$

V. Learning Context-Aware Latent Attribute Distribution

Based on the individual models trained for each attribute, ALM can provide service linkage recommendation, that is, given one API of interest, it can provide the link probability of any known API. However, given a mashup query context, not every attribute contributes the same. Take Fig. 4 as an example, two API nodes have nine attributes whose values are assigned by API L-LDA model. Our general ALM takes all attributes to calculate the link probability score, which is the production of corresponding value of each attribute from affinity distribution PDF. In contrast, we have developed a context-aware ALM learning model that takes into consideration the description of mashup and use only selected attributes and regard others as irrelevant. As shown in Fig. 4, five attributes are selected.

Each attribute of APIs will have an affinity distribution after the pre-training process. Our idea is to train a dedicated model to understand the different importance of the attributes to various mashup context, which we call attribute sensitivity to context. Like in the L-LDA model, we assume that attributes as

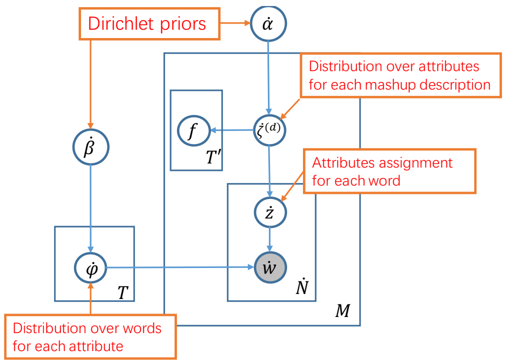


Attr_A	Attr_1	Attr_2	Attr_3	Attr_4	Attr_5	Attr_6	Attr_7	Attr_8	Attr_9
Value	0.322957	0.073508	0.744813	0.248181	0.318119	0.534449	0.149571	0.81126	0.553767

Attr_B	Attr_1	Attr_2	Attr_3	Attr_4	Attr_5	Attr_6	Attr_7	Attr_8	Attr_9
Value	0.481527	0.621531	0.761745	0.376721	0.747378	0.734616	0.097351	0.781369	0.331631

Affinity Distribution of Attr_1
0.20
0.15
0.10
0.05
0.00
1
0
1
Φ_1:Model trained for Attr_1
(a)
$P_{V_A,V_B} = \prod_{n=1}^{9} \Phi[Attr_{An}, Attr_{Bn}] = 0.10 * 0.21 * \cdots * 0.82$
Φ_2
Product of 9 attributes
$P_{V_A,V_B}^{Mashup\_\alpha} = \prod_{n=1}^{5} \Phi[Attr_{An}, Attr_{Bn}] = 0.10 * 0.32 * \cdots * 0.55$
Φ_3
Product of 5 attributes
Context-Based ALM Graph
$P_{V_A,V_B}^{Mashup\_\alpha}$
(b)
Φ_1 Φ_2 Φ_3 Φ_4 Φ_5 Φ_6 Φ_7 Φ_8 Φ_9
P_{V_A,V_B}
General ALM Graph

Fig. 4 Context-Aware ALM learning.



Dirichlet priors
$\dot{\alpha}$
Distribution over attributes for each mashup description
$\dot{\beta}$
f
T'
$\dot{\zeta}^{(d)}$
Attributes assignment for each word
$\dot{z}$
$\dot{\varphi}$
$\dot{w}$
T
$\dot{N}$
M
Distribution over words for each attribute

Fig. 5 Generative process of sensitivity probability of attributes.

labels are predefined and conditionally independent. To clarify, we call it mashup L-LDA model in the following sections. Given a mashup with description text, we aim to train a model to infer how related each attribute of APIs will contribute to linkage prediction. An attribute weight vector $\mathcal{F}^{(\mathrm{m})}$ is defined to indicate the importance of attributes of APIs to the context of mashup m. As summarized in Fig. 5, the generative process of attribute weight vector can be described as follows. The notations used are summarized in Table 2.

1) For each attribute of mashup $t = 1{:}T$
   Draw $\dot{\varphi}_t \sim Dirichlet(\dot{\beta})$
2) For each mashup description $m = 1{:}M$
   a) Draw $\dot{\zeta}^{(m)} \sim Dirichlet(\dot{\alpha})$
   b) For each word token $i = 1{:}\dot{N}$
      Draw an attribute $\dot{z}_i^{(m)} \sim Multinomial(\dot{\zeta}^{(\mathrm{m})})$
      Draw a word $\dot{w}_i^{(m)} \sim Multinomial(\dot{\varphi}_t)$
   c) Attribute weight vector:

$$\mathcal{F}^{(\mathrm{m})} = \left(f_1^{(m)}, f_2^{(m)}, \dots, f_{T'}^{(m)}\right) = \dot{\zeta}^{(\mathrm{m})},$$

$$where\ f_j^{(d)} \in [0,1], j \in T'$$

Assume that each mashup implies an attribute distribution $\dot{\zeta}^{(m)}$. Each word in the description is selected over an attribute-word distribution, after a specific attribute is selected. Learning process related to L-LDA is similar to the method mentioned in section IV.B thus will not be elaborated here.

Instead of considering all attributes equal, context-aware latent attribute learning would consider only attributes with their weight values larger than a predefined threshold or the top-k largest values:

$$P_{i,i'}^{m} = \prod_{f_t^m > threshold} \Phi_t\left[\tau_t^i, \tau_t^{i'}\right], t \in T'$$

Table 2. Notation used in context-aware latent attribute affinity distribution learning.

Symbol	Description
$\boldsymbol{M}$	Number of mashup descriptions
$\dot{\boldsymbol{N}}$	Number of words in one mashup description
$\boldsymbol{T}$	Number of attributes
$\boldsymbol{T'}$	Number of attributes
$\dot{\boldsymbol{\alpha}}, \dot{\boldsymbol{\beta}}$	The parameters of Dirichlet priors to the multinomial distribution $\dot{\zeta}^{(d)}$ and $\dot{\varphi}^{(Z)}$
$\dot{\boldsymbol{\varphi}}^{(\dot{z})}$	The parameters of multinomial distribution over words specific to attribute $\dot{z}$
$\dot{\boldsymbol{\zeta}}^{(m)}$	The parameters of multinomial distribution over attributes specific to description m
$\dot{\boldsymbol{z}}^{(\dot{n})}$	The attribute assign to word $\dot{n}$
$\dot{\boldsymbol{w}}$	The word token in mashup description
$\boldsymbol{f}_{\mathrm{t}}^{(m)}$	Attribute weight value specific to attribute t in context of mashup m

VI. Mashup Recommendation Technique

Now that we have described our two-layer framework, in this section, we will present our context-aware mashup recommendation technique. To ensure scalability and runtime performance, the training processes would be divided into offline and online phases, as shown in Fig. 6.

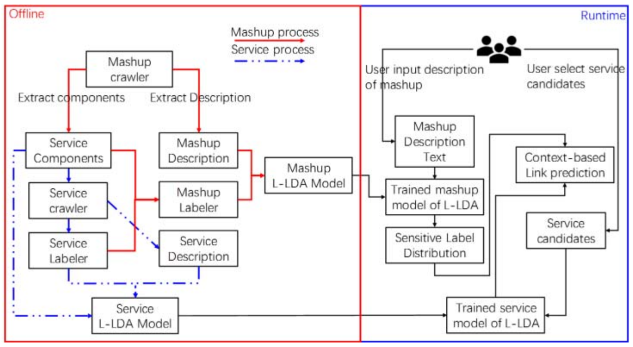


Fig. 6 Training of context-aware latent attribute distribution learning.

During the offline phase, known mashups information would be leveraged to extract their services components and descriptions through a mashup crawler. API components of mashups would be identified and analyzed through a service crawler to extract their descriptions and labels. Mashup labels comprise labels of their corresponding services.

Using API parameters and properties as its labels in the L-LDA training process, and service description from service providers as documents, the ALM linkage prediction model can generate a probability for any link between two APIs based on attributes values from the L-LDA model and attribute affinity distribution from the ALM model. Mashup L-LDA model will leverage the labels of the comprising API components in mashups, and mashup descriptions as document. Trained mashup L-LDA model could provide an attribute weight vector, given a new mashup description.

At runtime, a user can input natural language description of demanded mashup. The description will be put into the trained mashup model of L-LDA to calculate attribute weight vector, which indicates important attributes for linkage prediction based on the context of the mashup. Given the candidate services, the ALM model will provide the link probability between any user's API candidates and any known optional APIs, considering only sensitive attributes. The results will be shown to the users in the form of a ranked list of recommended services or yes-or-no threshold based prediction with information learned by the entire model.

VII. Experiments and Analysis

We have designed a collection of experiments to evaluate our ALM model, utilizing unstructured attributes to predict future affinity linkage.

A. Experimental Design on Real World Dataset

Each scientific publication typically contains a software implementation, in other words, a publication can be viewed as a special service (which is also the ultimate goal of the runnable paper initiative). Therefore, we treat a collection of scientific papers in the field of services computing as our testbed, where each paper represents a service node in a service social network. It is thus intuitive to model their topics as attributes, their content as service descriptions, the taxonomy of Services Computing as allowable labels, and the keywords in papers as pre-assigned labels. When a paper cites another one, it implies to exploit the program/software published in the other paper. In other words, (1) each paper represents an API; and (2) a paper also represents the context of a mashup that leverages other APIs (i.e., papers).

We crawled all ICWS (starting from 2003), SCC (starting from 2004), and TSC (starting from 2008) papers as our testbed. The total number of papers collected for experiments is about 2,500 and the number of internal citation relations within the papers is 1,140. We adopted the first-level entities in the taxonomy as the labels to feed the L-LDA model in Stanford Topic Modeling Toolbox[1]. For the testbed, we thus have 11 topics in total, which means the number of attributes for each paper is 11 in our ALM model. Terms that do not belong to the 11 topics were assigned to a special topic called 'others.'

Based on Formula (8), we tried to maximize the loglikelihood by selecting proper parameters for Bivariate Normal Distributions of 11 attributes, $\mathcal{N}(\mu_l, \Sigma_l), l = 1,2,3, \dots, l$. For each attribute, we tried to find the proper parameters to maximize the loglikelihood:

$$\mathcal{L}(\Theta_l) = \mathcal{L}(\mu_l, \Sigma_l) = Log\left(\prod_{A_{ij}=1} \theta_l[f_{il}, f_{jl}]\right) \qquad (9)$$

Normalized PDF value in [0,1] was used as factors to calculate the link probability for each pair of nodes based on Formula (1). For each pair of nodes that have no link connecting them, we can get a predicted value $\hat{A}_{ij}$:

$$\hat{A}_{ij} = \begin{cases} 1, & if\ P_{ij} > threshold \\ 0, & otherwise \end{cases}$$

Following the method described in section V, we used the context of the current paper as mashup description and infer its topic weight vector. From the vector, the topics with top three weights were selected for inferring the link probability between the current paper and any other candidates for citation. Because the number of topics for different papers varies, an average value of all PDF values was served as the final score. After normalization, the final score indicating the link probability falls into the interval $[0,1]$, which was also be used as the interval for threshold. In the training stage, we split the data samples randomly for 10-fold cross validation, and use 90% of samples for training and the rest 10% samples for testing. This strategy intends to better estimate the performance of our algorithms and to limit overfitting.

B. Computational Complexity

In this section, we discuss the computational complexity of our ALM method. We assume that the number of papers is D and the total number of topics is L, and there are N words in one paper.

For the L-LDA method, running one iteration of Gibbs Sampling takes $O(NL)$ steps. Since there are multiple papers, it takes $O(NLD)$ steps to scan all of them. To guarantee the convergence of probability distributions, it is necessary to run Gibbs sampler for sufficient number of iterations, which makes the overall running time to be $O(NKMI)$. Here I is the number of iterations. We used the parameters from L-LDA method as attributes vectors.

1) Evaluation

To evaluate the performance of our ALM model, several related algorithms were used as comparison. TF/IDF was used as a method to measure the importance of tokens in text and then to generate a term vector for each paper. Given term vectors of all papers, Cosine similarity was used to measure the distance of one paper to another. If the similarity between two papers is larger than a certain threshold, we assume it is very likely for them to form a link connecting each other in the future. For TF/IDF-Cosine method, the angle θ is considered to be the threshold to determine whether two papers have citation relation. To make the experiment result consistent, the threshold interval $[0, \pi/2]$ is projected to $[0,1]$.

The K-means algorithm was applied to cluster papers and it is then assumed that links would exist between papers in the same cluster. By tuning the number of clusters from 1 to 2,500, we expected the performance of prediction varying. To make the experiment result consistent, the cluster K interval $[1,2500]$ was projected to $[0,1]$.

We also used the parameters of Multinomial Distribution as vectors of papers to calculate the distance and used a threshold to determine if there would be a link in the future. Using such similarity measurement to infer link probability, different

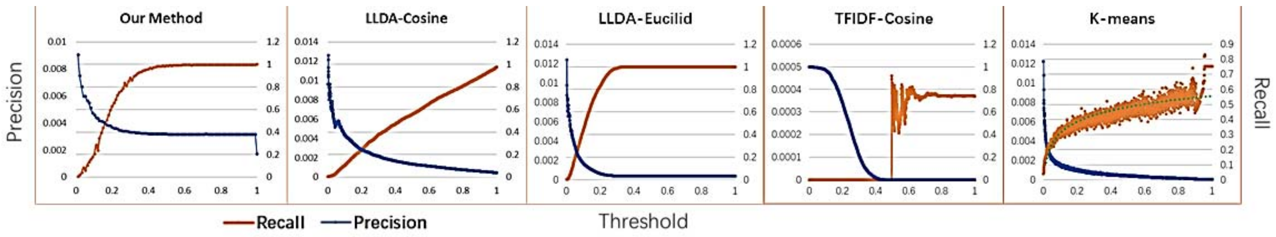


Fig. 7 Comparison of our method with state of the art approaches.

[1] https://nlp.stanford.edu/software/tmt/tmt-0.4/

distance measurements would lead to different precision and recall over all. Both Euclid metric and Cosine similarity metric were used for comparison.

Fig. 7 shows the precision and recall comparison between different algorithms. The ground truth is the true references of the test dataset. If one paper is predicted to refer to another paper and in fact it does refer, such a test sample will be considered as true positive.

2) *Results and Discussions*

It is necessary to consider that the probability for two nodes to form a link is not proportional to the similarity in value, but to the patterns of values of the two nodes we are viewing. Simple classification or clustering based on attributes values similarity is insufficient to capture the complex patterns when we consider a pair of nodes as one unit.

Methods like Cosine similarity, K-means clustering and distance calculation do not consider the historical linkage information. They consider only the attributes values and the similarity between two nodes. It can be observed from Fig. 7 that when the recall is higher than 0.4, the precision of our method is the highest, which means even after using parameters of Multinomial Distribution - the output from L-LDA model - our method still outperforms LLDA-Cosine, LLDA-Euclid and K-means. The results prove the merit of our method in fine-grained link prediction.

Some intermediate results deserve discussion as those reflect interesting patterns or latent information in reality. Fig. 8 shows the distribution of publication data samples of each attribute. The x-axis represents the referrer's value of each attribute, which, in the case of publications, is the value of each topic. The y-axis represents the referee's value of attributes. It can be noted that quite some data samples are distributed near the point (0,0), which indicates that when two papers have a citation relationship between them, it is likely that the attribute values for one topic of both papers are close to 0.

If one paper is related to a topic and the other paper is not at all related to it, we consider there should be another topic which contributes to the connection between two papers. Hence, it is cogent to remove those data samples during the training and prediction phases. The side effect of it, however, is that the data samples previously intended for training the link affinity distribution would be downscaled. That could intensify the issue of data sparsity.

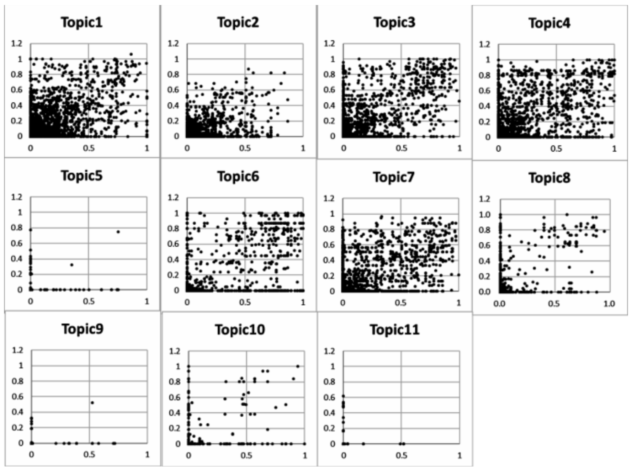


Fig. 8 Learned models for topics. The x-axis and y-axis denote the topic probability of API1 and API2, respectively.

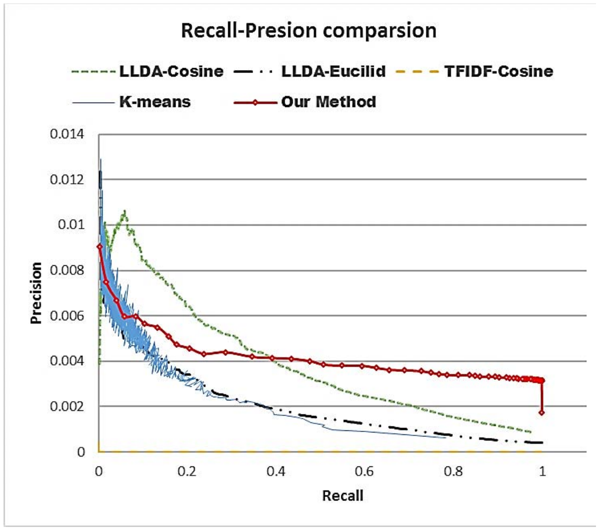


Fig. 9 Precision-recall impact comparison.

Fig. 9 is a precision-recall comparison graph of different algorithms. As we know precision and recall are typically inverse related and there is always a trade-off between them. It can be noted that the performance of our algorithm is the best compared to other algorithms under the condition that we consider the recall to be of more importance. This is because our evaluation criterion is that a link prediction algorithm should be capable of inferring unknown citation relationship based on the training data, without losing current information from history records. Given a test data set where the real citation relationship is known, the algorithm should have a relatively high recall, which indicating its ability of learning human research behavior and citing the reasonable paper as a part of references. The precision is not among evaluation criterion in that the truth set can only show what is the current paper citation information, but not what is the correct citation information. It is assumed that the current paper citation is not very complete due to limitation of cognition scope of researchers, which happens to be part of the motivation of our algorithm, to assist researchers in their study.

As shown in Fig. 9, the recall rates for most algorithms decrease rapidly as the precision increase. To reach the recall of 0.8, for instance, our algorithm remains its precision of more than 0.003. However, other algorithms only have a precision which is half of ours.

VIII. Conclusions

In this paper, we have demonstrated that fine-grained attribute-oriented model training can help predict the possibility of API linkage more accurately. The L-LDA model is used twice in our method: one is to extract information from unstructured data such as text description and papers, while the other one is to leverage latent information in the mashup context to provide context-aware predictions. Experimental results have proved that our model is able to prediction API linkage with a high recall.

In our future work, we plan to further study attribute modeling taking into consideration their dependency. Meanwhile, we plan to study the approximation approach for parameter estimation. Furthermore, we plan to apply our approach to a larger testbed for user study.

IX. Acknowledgment

This work is partially supported by National Aeronautics and Space Administration and National Science Foundation, under grants NNX16AB22G, NNX16AE15G, and NSF ACI-1443069.

X. References

[1] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, "Similarity Search for Web Services," in *Proceedings of 30th International Conference on Very Large Data Bases (VLDB)*, 2004, pp. 372–383.

[2] A.V. Paliwal, B. Shafiq, J. Vaidya, H. Xiong, and N. Adam, "Semantics-Based Automated Service Discovery," *IEEE Transactions on Services Computing (TSC)*, 5(2), 2012, pp. 260-275.

[3] Z. Zheng, H. Ma, M.R. Lyu, and I. King, "QoS-Aware Web Service Recommendation by Collaborative Filtering," *IEEE Transactions on Services Computing*, 4(2), 2011, pp. 140-152.

[4] J. Zhang, W. Tan, J. Alexander, I. Foster, and R. Madduri, "Recommend-As-You-Go: A Novel Approach Supporting Services-Oriented Scientific Workflow Reuse," in *Proceedings of IEEE International Conference on Services Computing (SCC)*, Washington DC, USA, 2011, pp. 48-55.

[5] Y. Zhong, Y. Fan, K. Huang, W. Tan, and J. Zhang, "Time-Aware Service Recommendation for Mashup Creation," *IEEE Transactions on Services Computing (TSC)*, 8(3), 2015, pp. 356-368.

[6] D.M. Blei, A.Y. Ng, and M.I. Jordan, "Latent Dirichlet Allocation," *Journal of Machine Learning Research*, vol. 3, 2003, pp. 993-1022.

[7] Z. Gao, Y. Fan, C. Wu, W. Tan, J. Zhang, Y. Ni, B. Bai, and S. Chen, "SeCo-LDA: Mining Service Co-occurrence Topics for Recommendation," in *Proceedings of IEEE International Conference on Web Services (ICWS)*, San Francisco, CA, USA, 2016, pp. 25-32.

[8] S. Lee, L. Pan, C. Zhai, B. Tang, T. Kubar, J. Zhang, and W. Wang, "Climate Model Diagnostic Analyzer," in *Proceedings of IEEE International Conference on Big Data (Big Data)*, Santa Clara, CA, USA, 2015, pp. 1887-1891.

[9] M. Kim and J. Leskovec, "Multiplicative Attribute Graph Model of Real-World Networks," in *Proceedings of International Workshop on Algorithms and Models for the Web-Graph*, Stanford, CA, USA. 2010, pp. 62-73.

[10] D. Liben-Nowell and J. Kleinberg, "The Link-Prediction Problem for Social Networks," *Journal of the American Society for Information Science and Technology*, 58(7), 2007, pp. 1019-1031.

[11] M.A. Hasan, V. Chaoji, S. Salem, and M. Zaki, "Link Prediction Using Supervised Learning," in *Proceedings of Workshop on Link Analysis, Counter-Terrorism and Security* (SDM), 2006.

[12] M. Kim and J. Leskovec, "Modeling Social Networks with Node Attributes Using the Multiplicative Attribute Graph Model," in *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence,* 2011, pp. 400-409.

[13] N.Z. Gong, A. Talwalkar, L. Mackey, L. Huang, E.C.R. Shin, E. Stefanov, E.R. Shi, and D. Song, "Joint Link Prediction and Attribute Inference Using A Social-Attribute Network," *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(2), 2014, pp. 27.

[14] K. Palla, Z. Ghahramani, and D.A. Knowles, "An Infinite Latent Attribute Model for Network Data," in *Proceedings of the 29th International Conference on Machine Learning*, 2012, pp. 1607-1614.

[15] W.L. Buntine, "Operations for Learning with Graphical Models," *Journal of Artificial Intelligence Research*, 1994.

[16] Thomas L Griffiths and M. Steyvers, "Finding Scientific Topics," in *Proceedings of the National Academy of Sciences*, vol. 101, supp. 1, 2004, pp. 5228-5235.

[17] D. Ramage, D. Hall, R. Nallapati, and C.D. Manning, "Labeled LDA: A Supervised Topic Model for Credit Attribution in Multi-Labeled Corpora," in *Proceedings of The Conference on Empirical Methods in Natural Language Processing*, 2009, pp. 248-256.