SRaSLR: A Novel Social Relation Aware Service Label Recommendation Model

Yeqi Zhu*, Mingyi Liu*, Zhiying Tu†, Tonghua Su*, Zhongjie Wang*
*Faculty of Computing, Harbin Institute of Technology, Harbin, China
†School of Computer Science and Technology, Harbin Institute of Technology (Weihai)
1173710229@stu.hit.edu.cn, {liumy, tzy_hit, thsu, rainy}@hit.edu.cn

***Abstract*—With the rapid development of new technologies such as cloud, edge and mobile computing, the number and diversity of available services are dramatically exploding and services have become increasingly important to people's daily work and life. As a consequence, using service label recommendation techniques to automatically categorize services plays a crucial role in many service computing tasks, such as service discovery, service composition, and service organization. There have been many service label recommendation studies that have achieved remarkable performance. However, these studies mainly focus on using the text information in service profiles to recommend labels for services while overlooking those social relations that widely exist among services. We argue that such social relations can help to obtain more precise recommendation results. In this paper, we propose a novel Social Relation aware Service Label Recommendation model called SRaSLR, which combines text information in service profiles and social network relations among services. A deep learning based model is constructed based on feature fusion of the two perspectives. We conduct extensive experiments on the real-world *ProgrammableWeb* dataset, and the experiment results show that SRaSLR yields better performance than existing methods. Additionally, we discuss how service social network affects service label recommendation performance based on the experiment results.**

***Index Terms*—Web-based Services; Service Label Recommendation; Service Social Network; Feature Fusion; Deep Learning**

I. Introduction

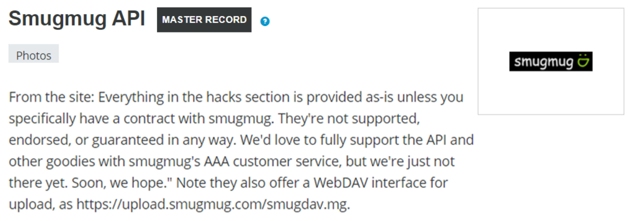
With the rapid development of new technologies such as cloud, edge and mobile computing, there have appeared more and more web-based services which are publicized by different organizations, delivered in different regions, and offer a wide range of functionalities. As a consequence, service label recommendation has become a new research hotspot in services computing community. Automatic service label recommendation can bring significant benefits to users, developers and platforms. For users, they can quickly and accurately find exact services that satisfy their personalized requirements by keywords from well-classified web services [1]. For developers, they can easily find suitable constituent services for their coarse-grained applications, thus reducing their burden of development and increasing efficiency. For platforms, they can save a lot of labor costs in indexing and organizing services that grow rapidly. Additionally, service label recommendation can make services more likely to be pushed to potential users, which leads to a wider range of usage and ultimately advances the continuous improvement of services.

There have been many studies that focus on service label recommendation and have achieved remarkable performance, such as WTLearning [2], LDA-SVM [3] and ServeNet [4], [5]. These studies are mainly based on text information in service profiles, such as service descriptions and service names. However, texts do not always contain enough information for significantly distinguishing different categories of services. For example, as descriptions of services on *ProgrammableWeb*[1] are provided by their uploaders (not necessarily their service providers), it requires that uploaders are familiar with the services and are willing to take time to elaborately compose the descriptions. Actually, many service descriptions are too brief or vague to provide enough information for humans to determine their categories. Taking the description of a service *Smugmug* as an example (shown in Fig. 1a), the text looks long enough, but there are no relevant words that can be associated with the relevant category *Photos* that the service belongs to. Another example is the description of *Pixelpipe* (shown in Fig. 1b): the text is so brief that it is impossible to obtain any information to infer the category it belongs to.

There are *invocation* and *dependency* relations between services, and these relations make services naturally constitute a service social network [6]. Typical phenomena in social networks are still present in service social networks, e.g., homophily [7] which means that individuals tend to associate and bond with other similar ones. Fig. 2 shows a part of a service social network centered on *Smugmug* and *Pixelpipe*, where more than 40% of services (green nodes) are using *Photos* as their primary label. And it is clear that services in the category *Photos* are closer to *Smugmug* and *Pixelpipe* than those in other categories. Therefore, service social networks provide useful category-related information that texts haven't. We conjecture that taking service social networks into account can complement text descriptions of services, thus improving the accuracy of service label recommendation.

In this paper, we propose a novel Social Relation aware Service Label Recommendation model named SRaSLR. Compared with the traditional methods, SRaSLR considers not only text information but also the information in service social network. We fuse text features and social relation features to

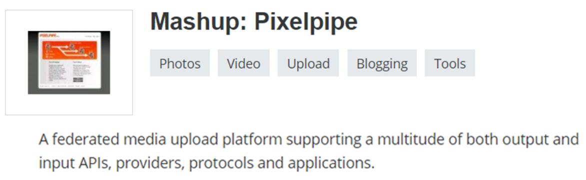
[1]https://www.programmableweb.com


Smugmug API MASTER RECORD

Photos

From the site: Everything in the hacks section is provided as-is unless you specifically have a contract with smugmug. They're not supported, endorsed, or guaranteed in any way. We'd love to fully support the API and other goodies with smugmug's AAA customer service, but we're just not there yet. Soon, we hope." Note they also offer a WebDAV interface for upload, as https://upload.smugmug.com/smugdav.mg.

(a) An example of vague service description


Mashup: Pixelpipe

Photos Video Upload Blogging Tools

A federated media upload platform supporting a multitude of both output and input APIs, providers, protocols and applications.

(b) An example of brief service description

Fig. 1: Examples of vague or brief service descriptions

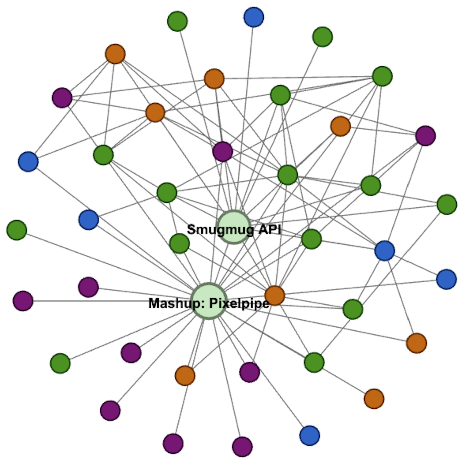


Fig. 2: A part of service social network centered on *Smugmug* and *Pixelpipe*

achieve better performance of service label recommendation. Experiments conducted on the real-world *ProgrammableWeb* dataset show that SRaSLR achieves better performance on both top-5 accuracy and top-1 accuracy than recent state-of-the-art methods. In addition, we conduct additional experiments to explore the impact of different service social network representations and service description length on the service label recommendation task. We have open sourced all the source code and data on GitHub[2] for researchers who are interested in service label recommendation research.

The remainder of the paper is organized as follows: Section II introduces the related works. Section III explains key definitions used in this paper. Section IV introduces SRaSLR. Section V introduces the dataset, experimental settings, and evaluation metrics. Section VI demonstrates and analyzes the experimental results, along with further discussions. Section VII concludes the paper and proposes the future work.

II. Related Works

Current studies often treat web service label recommendation problem as a text classification problem, which is a classical Natural Language Processing problem aiming at assigning labels to textual units. To incorporate service social network features, Graph embedding methods are also needed for learning low-dimensional latent representation of nodes in a graph [8].

A. Text Classification based Service Label Recommendation

Methods for text classification based service label recommendation can be categorized into the following two categories:

- **Machine learning methods** for service label recommendation task have achieved good results in recent years [9]–[11], Most classical approaches in this category have several limitations: (1) they depend heavily on feature engineering and this would result in higher difficulty and cost; (2) they cannot make full use of large amounts of training data because the features have been pre-defined.
- **Deep learning approaches** transform the low-level texts into high-level representations without manual feature engineering [12]. For example, Recurrent Neural Network (RNN) based models [13] view a text, e.g., service description as a sequence of words and can capture word dependencies and text structures [14], [15] accurately; Long Short-Term Memory (LSTM) is one of the most popular RNN-based architectures [16] that are inclined to learn features across time; Convolutional Neural Network (CNN) tends to recognize patterns across space [12]. Additionally, transformer-based Pre-trained Language Models (PLMs) [17] use much deeper network architectures and pre-trained models on extremely large datasets [12]. Bert [18] is one of the most widely used PLM models, and text classification accuracy has reached a new level since it was publicized. Some approaches make use of multiple models to make service label recommendation, such as ServeNet [5] combines Bert, CNN and LSTM to capture both local and global features, PSO-Bi-LSTM-CNN model [19] integrates particle swarm optimization with the Bi-LSTM and CNN.

B. Graph Embedding

Graph embedding also has a breakthrough in the past years. How to extract features from nodes and their neighboring

[2]https://github.com/HIT-ICES/SRaSLR

nodes and edges effectively is the kernel problem. The key point is to design an embedding algorithm, which can be classified as follows:

- **Factorization-based**. The basic idea is dimensionality reduction. Locally Linear Embedding (LLE) [20] is one of the methods in this category, and its kernel idea is that a node can be composed with its neighbors by the linear weighted sum method.
- **Random walk-based**. These methods firstly walk from node to node and generate walk sequences. Then they view the walk sequences as text sequences and then use Word2Vec [21] to process these sequences. DeepWalk [22] is a representative method in this category. It tends to learn a partial structure from a graph with a completely random search. Node2Vec [23] is a biased random walk method which can switch the search strategy such as Breadth First Search (BFS) and Depth First Search (DFS) by changing the arguments of the model.
- **Deep learning-based**. For example, Structural Deep Network Embedding (SDNE) [24] is a semi-supervised learning method. It uses an autoencoder to optimize the similarity of two connected nodes and neighborhood similarity for any two nodes together. Vectors learned by this method can represent both local and global structural features, as well as keeping robustness for sparse networks.

III. Definitions

In this section, we explain some definitions used in paper, including **Services**, and how they compose **service social network**, **A-A network**, **M-M network**, and the task of **service label recommendation**.

Definition 1: **A service** is a web-based atomic API or the composition of multiple APIs (called a mashup [25]) that offer specific functionality to clients. A collection of available services is denoted as $S = \{s|s \in M \cup A\}$, where $M = \{m_1, m_2, \ldots, m_p\}$ and $A = \{a_1, a_2, \ldots, a_q\}$ is the set of mashups and the set of APIs, respectively. For each service $s_i \in S$, there is a primary label l_i indicating its category. The label space is fixed and can be represented as $L = \{1, 2, \ldots, K\}$. Each service s_i has text descriptions (i.e., its profile) denoted as d_i.

Service social network can be represented in many forms. The most straightforward one is a network generated by the invocation relations between mashups and APIs, which is also called Mashup-API network.

Definition 2: **Mashup-API network (M-A network)** is an undirected unweighted bipartite graph and can be denoted as $G_{MA} = \{S, E_{MA}\}$. $E_{MA} = \{(u, v)|u \in M, v \in A\}$ denotes the invoking relations between mashups and APIs.

M-A network is a bipartite graph, which makes it difficult to directly demonstrate the relations between APIs or between mashups, especially the strength of relations between different APIs/mashups.

Definition 3: **API-API network (A-A network)** is a homogeneous network, which is generated by an M-A network G_{MA}, and can be denoted as $G_{AA} = \{A, E_{AA}\}$. $E_{AA} = \{(u, v, w)|u, v \in A\}$ represents the API concurrence relations, and w is the weight of edge (u, v), representing the times they co-occur.

Definition 4: **Mashup-Mashup network (M-M network)** is similar to A-A network that can be denoted as $G_{MM} = \{M, E_{MM}\}$ where $E_{MM} = \{(u, v, w)|u, v \in M\}$ is the mashup concurrence relations and w is the weight of edge (u, v), which is the number of common APIs used by these mashups.

With the M-M network and A-A network, it would be easier to demonstrate the relations and strength between APIs or mashups. Fig. 3 shows an example of transformation between different service social networks.

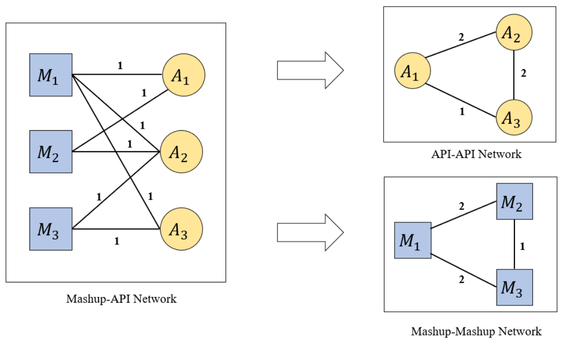


Fig. 3: Transformation between different types of service social networks

Definition 5: **Service label recommendation** is a task to train a model that can predict the primary label $l_i \in L$ for a given service $s_i \in S$. This process can be formally represented as $l_i = \text{argmax}P(l|s_i, d_i)$.

IV. Architecture of SRaSLR

Fig. 4 shows the architecture of SRaSLR. The input of the model is the ID of a service s and its corresponding description d. The main body of the architecture is composed of an encoder and a decoder. The encoder is responsible for encoding the service text information and service social network information into feature vectors, and the decoder is responsible for mapping the encoded feature vectors to the label space L. The rest of this section describes the encoder, decoder, and optimization objective in detail.

A. Encoder

The encoder consists of three parts: service description feature encoding, service social network feature encoding and feature fusion.

Service description feature encoding: This part uses Bert to embed service description to a vector representation, which can well capture text contextual features. The output of Bert consists of two modes, one is sequential and the other is pooled. Here, we use the pooled output to represent the

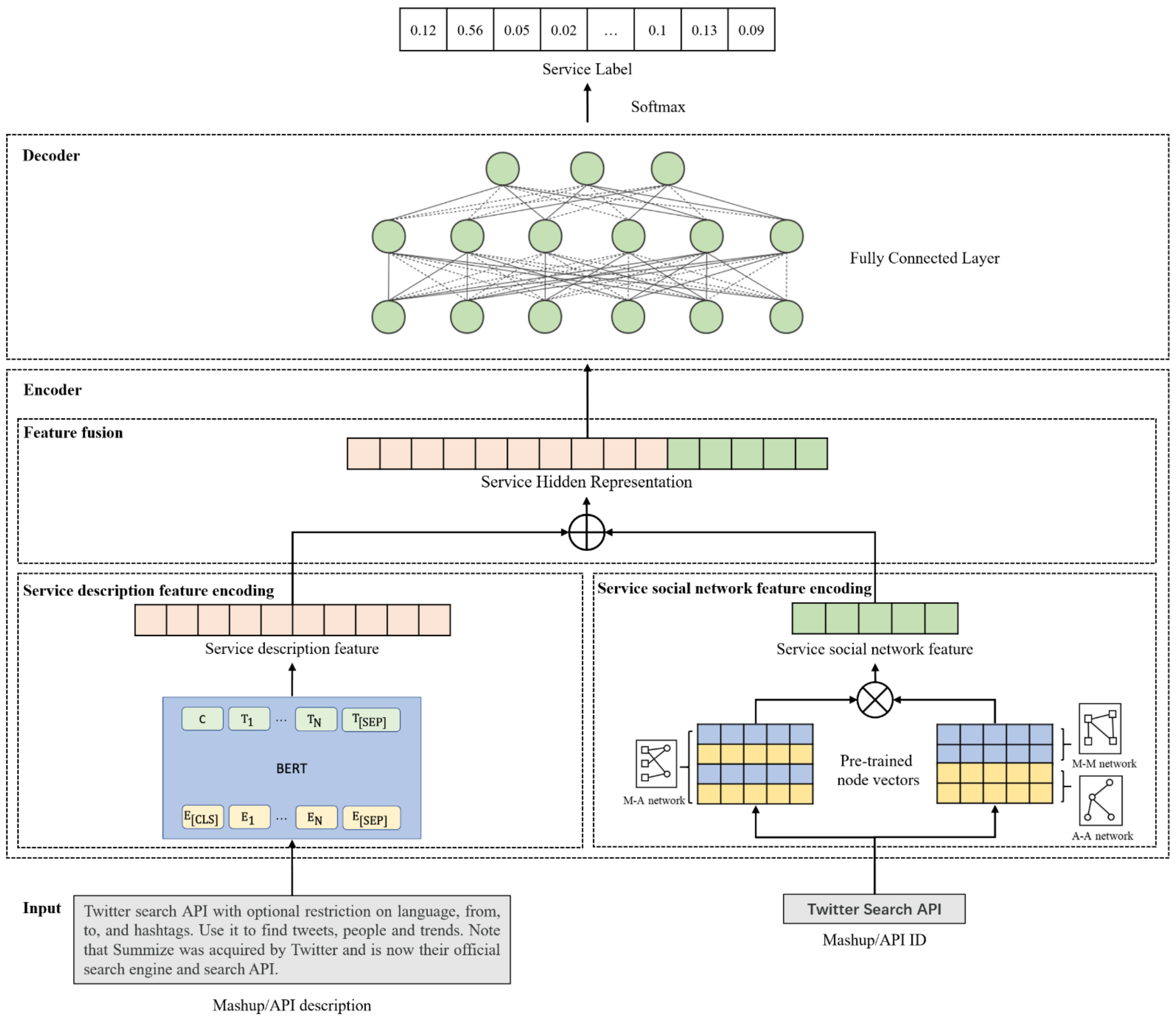

0.12
0.56
0.05
0.02
…
0.1
0.13
0.09
Service Label
Softmax
Decoder
Fully Connected Layer
Encoder
Feature fusion
Service Hidden Representation
Service description feature encoding
Service description feature
C
T1
TN
T[SEP]
BERT
E[CLS]
E1
EN
E[SEP]
Service social network feature encoding
Service social network feature
Pre-trained node vectors
M-A network
M-M network
A-A network
Input
Twitter search API with optional restriction on language, from, to, and hashtags. Use it to find tweets, people and trends. Note that Summize was acquired by Twitter and is now their official search engine and search API.
Mashup/API description
Twitter Search API
Mashup/API ID

Fig. 4: Architecture of SRaSLR

description features. For a given service description d, Bert converts it into a vector $\boldsymbol{x}^d$ with fixed-length D_d:

$$\boldsymbol{x}^d = f_{bert\_pooled}(d) \tag{1}$$

Service social network feature encoding: This part firstly uses Node2Vec to pre-train three embedding matrices $X_{MA} \in R^{|S|\times D_n}$, $X_{MM} \in R^{|M|\times D_s}$, and $X_{AA} \in R^{|A|\times D_n}$ for M-A, M-M, and A-A networks independently. X_{AA} and X_{MM} are concatenated to represent the complete service social network embedding matrix:

$$X_{MM\_AA} = \begin{bmatrix} X_{MM} \\ X_{AA} \end{bmatrix} \tag{2}$$

Then each service s has two corresponding node embeddings $\boldsymbol{n_1} = [n_{1,1}, n_{1,2}, \ldots, n_{1,D_n}]$ and $\boldsymbol{n_2} = [n_{2,1}, n_{2,2}, \ldots, n_{2,D_n}]$ coming from X_{MA} and X_{MM_AA}, respectively, and feature representation of s in service social network requires fusion of these two node embeddings. One approach is to concatenate the two node embeddings:

$$\begin{aligned} \boldsymbol{x}^s_{concatenate} &= (\boldsymbol{n_1}, \boldsymbol{n_2}) \\ &= [n_{1,1}, \ldots, n_{1,D_n}, n_{2,1}, \ldots, n_{2,D_n}] \end{aligned} \tag{3}$$

and another approach is to make a hadamard production [26] of the two node embeddings:

$$\begin{aligned} \boldsymbol{x}^s_{hadamard} &= \boldsymbol{n_1} \odot \boldsymbol{n_2} \\ &= [n_{1,1}n_{2,1}, n_{1,2}n_{2,2}, \ldots, n_{1,D_n}n_{2,D_n}] \end{aligned} \tag{4}$$

We use $\boldsymbol{x}^s$ to denote the service social network feature (might be $\boldsymbol{x}^s_{concatenate}$ or $\boldsymbol{x}^s_{hadamard}$). Note that the dimension of $\boldsymbol{x}^s$ is D_s.

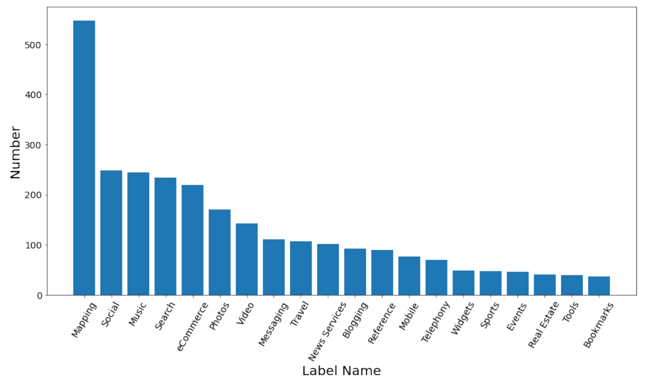


Fig. 5: Frequency distribution of top-20 most common labels

Feature fusion: This part fuses service description feature $\boldsymbol{x}^d$ and service social network feature $\boldsymbol{x}^s$ into a final feature $\boldsymbol{x} \in R^D$ by concatenation:

$$\boldsymbol{x} = (\boldsymbol{x}^d, \boldsymbol{x}^s) \tag{5}$$

B. Decoder

The decoder starts with two fully connected layers:

$$\begin{aligned} \boldsymbol{h} &= \boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1 \\ \boldsymbol{z} &= \boldsymbol{W}_2\boldsymbol{h} + \boldsymbol{b}_2 \end{aligned} \tag{6}$$

where $\boldsymbol{W}_1 \in R^{D\times D}$ and $\boldsymbol{W}_2 \in R^{K\times D}$ are weights, and $\boldsymbol{b}_1 \in R^D$ and $\boldsymbol{b}_2 \in R^K$ are biases.

Then a softmax layer is used to transform the hidden state $\boldsymbol{z}$ into category probabilities:

$$\hat{y}_k = \frac{\exp z_k}{\sum_{j=1}^{K} \exp z_j} \tag{7}$$

C. Optimization Object

SRaSLR adopts the cross-entropy as the loss function which can be described as:

$$loss = -\sum_{j}^{K} I(l,j)log(\hat{y}_j) \tag{8}$$

where l is the real label of the service, and $I(\cdot)$ is an indicator function.

V. Experiment Setup

In this section, we introduce the dataset, experimental settings and evaluation metrics.

A. Dataset

We collected a total of 23,520 APIs and 7,947 mashups from *ProgrammableWeb*. A set of steps are conducted to clean up the dataset, which can be summarized as follows:

S1 Remove the services without description or labels, as they cannot be trained or predicted.

S2 Remove the mashups that consist of less than 2 APIs, and remove the APIs that are invoked by mashups less than 2 times, because these APIs or mashups are nearly isolated in the service social network.

S3 Select the primary category as the label of each service.

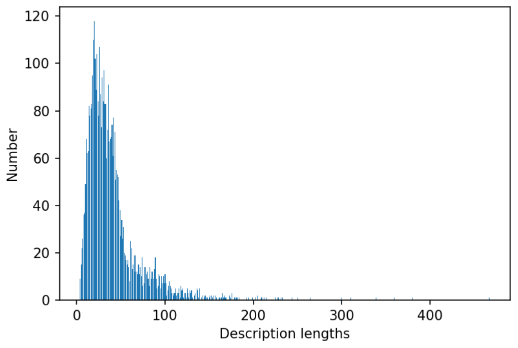


Fig. 6: Distribution of service description length

After the cleanup, the dataset contains 4,099 services, including 3,380 mashups and 719 APIs covering 255 labels. Fig. 5 shows the frequency distribution of top-20 most common labels. It is obvious that the dataset is imbalanced. Fig. 6 shows the distribution of the length of service descriptions, which ranges from 1 to 425 words. There are about 11.4% of service descriptions that are shorter than 15 words and 99.9% of service descriptions that are less than 300 words. Table I shows some basic statistics of three types of service social networks.

TABLE I: Statistics of M-A, M-M and A-A networks

	Average degree	Weighted degree	Density	Average path length
M-A	5.222	5.222	0.001	3.876
M-M	5.376	5.909	0.002	5.480
A-A	9.531	20.344	0.013	3.037

B. Experimental Settings

A total of 12 methods are used for comparisons in our experiments, including 4 text-only methods (LSTM, TextCNN, ServeNet and Bert-Baseline), 4 service social network only methods (MA, MM-AA, ALL-Cat, ALL-Hadamard), and 4 SRaSLR variations which use different service social network features in the encoder, including MA, MM-AA, ALL-Cat and ALL-Hadmard where MA and MM-AA apply X_{MA} and X_{MM_AA} as their node embedding matrix, respectively, and ALL-Cat and ALL-Hadamard use concatenation and hadamard production as their fusion methods, respectively.

We conduct 5 independent experiments for each method to prevent serendipity, and early-stop is applied to avoid over-fitting. All the experiments are conducted on a device with GPU Titan XP running Ubuntu 18.10. Source code is implemented based on PyTorch[3].

The dataset is randomly divided into training set, validation set, and test set by 6:2:2. The training batch size is set as 16.

[3]https://pytorch.org/

For the text feature encoding part, LSTM and TextCNN use a $128d$ Word2Vec [21] word embedding trained on *text8*[4]. Other methods use the *bert-base-uncased* provided by Transformers [27] as the text encoder, where *MAX_LENGTH* is set to 300 and the output vector dimension is 768. For the service social network encoding part, the dimension of node vectors D_n is set to 100. *walk_length* is set as 5 for the Node2Vec[5] and the other arguments remain default. In order to avoid overfitting, the dropout method [28] is applied and the rate is set to 0.5 for the fully-connected layer in the decoder.

The Adam [29] optimizer is used to minimize the objective function. To avoid insufficient learning, different learning rates are set for different layers. The learning rate of Bert is 0.00003 for fine-tuning, while other layers use a learning rate 0.001.

C. Evaluation Metrics

Top-k accuracy is used to evaluate a classification model by calculating credits if the right answer appears in the top k predictions. Top-1 accuracy is often used to evaluate a binary classification model and top-5 accuracy is often used to evaluate a multi-class classification model which usually requires the number of classes $\geq$ 10. For a given service s_i, SRaSLR predicts the label probabilities $\hat{\boldsymbol{y}}_{\boldsymbol{s_i}}$. The top-5 accuracy on the dataset is:

$$Acc^{top5} = \frac{|\{s_i | s_i \in S \land l_i \in \hat{\boldsymbol{y}}_{\boldsymbol{s_i}}{}^{top5}\}|}{|S|} \tag{9}$$

where $\hat{\boldsymbol{y}}_{\boldsymbol{s_i}}{}^{top5}$ denotes the top 5 possible labels in the prediction for service s_i.

VI. Experimental Results And Discussion

In this section, we firstly analyze the experimental results, and then discuss and explain some phenomena appearing in the experiment. Table II shows the experimental results.

A. Results

1) Performance: Among the 12 comparison methods, our proposed SRaSLR obtains the best and second-best performance in both top-1 accuracy and top-5 accuracy. Compared with best text-only (service social network-only) method, SRaSLR improves top-1 accuracy and top-5 accuracy by **0.5% (6.3%)** and **3.1% (9.2%)**, respectively. Since text-only methods use a large amount of external data to pre-train a language model as text feature encoder, so the performance of text-only methods (except LSTM) perform better than service social network-only methods. In the text-only methods, Bert-Baseline and ServeNet are significantly better than LSTM and TextCNN because of the use of Bert as text feature encoder with more parameters and pre-training data. It is not surprising that the performance of the recently proposed state-of-the-art model ServeNet is significantly worse than that of Bert-Baseline. The reasons for the performance degradation in ServeNet might be: (1) the unnecessary CNN and LSTM layers corrupt the already well-represented service description feature vectors; (2) the simple addition of the service name feature vectors and the service description feature vectors brings a negative impact to the model. Therefore, we consider that the improvement of ServeNet over traditional methods comes from the pre-trained language model Bert rather than from the superiority of its model architecture.

MA achieves the best top-1 accuracy and second-best top-5 accuracy among the network-only approaches due to its ability of clustering services having similar functionalities. ALL-Cat's top-5 accuracy is **0.6%** higher than MA but top-1 accuracy is **2.9%** lower. MM-AA does not live up to expectations, and the reason will be discussed in detail in Section VI-B1. ALL-Hadamard's top-1 and top-5 accuracy are **8.0%** and **11.7%** lower than ALL-Cat's, respectively, indicating that fusing the service social network feature vectors from different spaces by hadamard product cannot represent the service features properly without the assistance of other features.

SRaSLRs have the best top-1 and top-5 accuracy in the experiment because they can fuse the text features and the social network features. SRaSLR(ALL-Cat) makes full use of text and service social network, so it achieves the highest top-1 and top-5 accuracy. The MM-AA has a different performance if it works with text features, which is mainly because the MM-AA provides direct relations among mashups or APIs and Bert provides the text features that can help build up the relations between mashups and APIs. Therefore, SRaSLR(MM-AA) is slightly better than SRaSLR(MA). SRaSLR(ALL-Hadamard) is also better than SRaSLR(MA) and SRaSLR(MM-AA) on the top-1 accuracy. Section VI-B2 explains how the service social network in SRaSLR affects the recommendation results.

2) Efficiency: Network-only approaches and TextCNN have the fastest training speed (batches per second) since they have much fewer parameters than the other methods. LSTM also has a small number of parameters, but its recursive structure leads to a slower training speed. ServeNet is the slowest method, and its speed is only 69% of Bert-Baseline because it uses Bert to process descriptions and names simultaneously, which requires to update the parameters of Bert twice during the backpropagation process, and this costs a lot of time. Though SRaSLR is fine-tuning Bert and node vectors together, SRaSLR does not introduce complex layers, so it can still maintain a fast training speed in the training process.

The number of epoch when a model is to converge implies whether the optimization process is stable. Compared with other methods (except LSTM) whose epoch numbers to converge fluctuate, SRaSLRs come to converge after training for only 5 epochs, indicating they have good stability. The assistance of network and different learning rates that are properly set for both Bert and other parameters assure a very stable optimization in SRaSLRs.

B. Discussion

Here we analyze the performance difference between SRaSLRs based on different types of networks, and analyze

[4]http://mattmahoney.net/dc/text8.zip
[5]https://github.com/eliorc/node2vec

TABLE II: Experiment Results of Text-only, Network-only and Fusion Methods

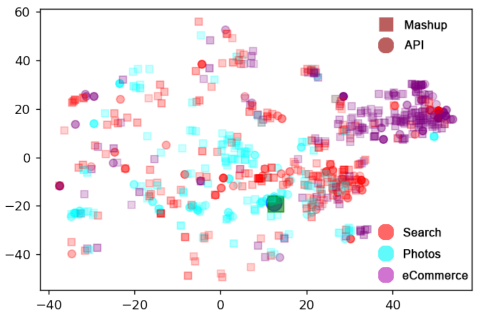
		Text				Service social network				Text + Service social network			
		LSTM	TextCNN	ServeNet[a]	Bert-Baseline	MA	MM-AA	ALL-Cat[b]	ALL-Hadamard	SRaSLR (MA)	SRaSLR (MM-AA)	SRaSLR (ALL-Cat)	SRaSLR (ALL-Hadamard)[c]
Top-1	Round 1	0.134	0.290	0.328	0.357	0.306	0.117	0.262	0.201	0.350	0.378	0.360	0.368
	Round 2	0.134	0.285	0.330	0.361	0.299	0.107	0.279	0.182	0.340	0.355	0.361	0.359
	Round 3	0.134	0.290	0.344	0.352	0.298	0.107	0.276	0.199	0.339	0.359	0.361	0.365
	Round 4	0.134	0.295	0.350	0.362	0.301	0.123	0.252	0.187	0.346	0.340	0.360	0.356
	Round 5	0.134	0.299	0.294	0.356	0.296	0.118	0.284	0.185	0.361	0.362	0.373	0.350
	Avg (Std)[d]	0.134 (±0.000)	0.292 (±0.005)	0.329 (±0.022)	0.358 (±0.004)	0.300 (±0.004)	0.115 (±0.007)	0.271 (±0.013)	0.191 (±0.009)	0.347 (±0.009)	0.359 (±0.014)	**0.363 (±0.006)**	0.360 (±0.007)
Top-5	Round 1	0.373	0.550	0.590	0.623	0.541	0.335	0.545	0.446	0.629	0.621	0.637	0.620
	Round 2	0.373	0.540	0.554	0.601	0.535	0.315	0.539	0.417	0.630	0.634	0.643	0.618
	Round 3	0.373	0.549	0.589	0.610	0.540	0.345	0.543	0.433	0.626	0.620	0.648	0.632
	Round 4	0.373	0.557	0.582	0.606	0.540	0.339	0.537	0.427	0.599	0.630	0.626	0.615
	Round 5	0.373	0.540	0.573	0.587	0.534	0.346	0.559	0.413	0.642	0.635	0.626	0.624
	Avg (Std)	0.373 (±0.000)	0.547 (±0.007)	0.578 (±0.015)	0.605 (±0.013)	0.538 (±0.003)	0.336 (±0.013)	0.544 (±0.009)	0.427 (±0.013)	0.625 (±0.016)	0.628 (±0.007)	**0.636 (±0.010)**	0.622 (±0.007)
Training Speed	Round 1	90.11	155.38	1.63	2.29	156.90	153.89	145.95	177.65	2.38	2.33	2.36	2.34
	Round 2	91.59	151.10	1.62	2.33	156.56	150.02	177.21	156.66	2.38	2.28	2.39	2.39
	Round 3	91.62	152.53	1.64	2.38	153.12	158.76	176.88	143.22	2.29	2.31	2.29	2.24
	Round 4	92.21	152.10	1.63	2.39	156.56	143.32	172.56	177.55	2.31	2.38	2.38	2.32
	Round 5	93.23	150.06	1.65	2.38	156.54	185.62	177.21	155.78	2.25	2.34	2.29	2.33
	Avg (Std)	91.75 (±1.13)	152.23 (±2)	1.63 (±0.01)	2.35 (±0.04)	155.94 (±1.58)	158.32 (±16.27)	169.96 (±13.57)	162.17 (±15.05)	2.32 (±0.06)	2.33 (±0.04)	2.34 (±0.05)	2.32 (±0.05)
Convergence Epoch	Round 1	3	10	6	5	9	10	7	10	5	5	5	5
	Round 2	3	10	7	9	10	11	7	11	5	5	5	5
	Round 3	3	8	7	7	9	10	7	10	5	5	5	5
	Round 4	3	10	7	6	10	10	7	10	5	5	5	5
	Round 5	3	10	6	6	9	9	6	11	5	5	5	5
	Std	±0.00	±0.90	±0.55	±1.52	±0.55	±0.70	±0.48	±0.55	±0.00	±0.00	±0.00	±0.00

[a] ServeNet is reproduced according to the paper [5]

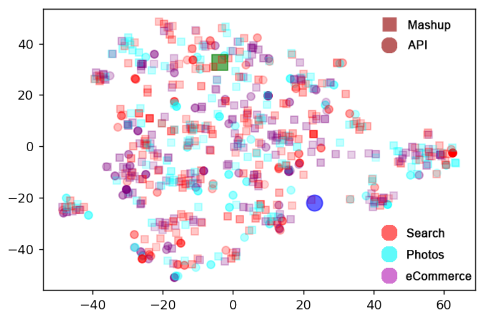
[b] MA and MM-AA apply X_{MA} and X_{MM_AA} as their node embedding matrix, respectively. ALL-Cat and ALL-Hadamard use both and the fusion methods are concatenation and hadamard production, respectively

[c] SRaSLR(*) means SRaSLR applies * in the service social network encoding part

[d] Bold font indicates the best, and underline indicates the second best



(a) Service social network features X_{MA}



(b) Service social network features X_{MM_AA}

Fig. 7: Visualization of two types of service social network features

the difference between the contributions made by text features and service social network features.

1) Difference between MA and MM-AA: Experimental results show that the MA network achieves the highest top-1 and top-5 accuracy when only service social networks are used to make predictions, but MM-AA's performance is not satisfying. An explanation is that the MA network contains completed invocation relations and all nodes are trained together, but the MM network and AA network are trained independently and later their vectors are put together; this results in the loss of invocation relations and consequently, unsatisfying performance. Fig. 7 shows the visualization of some service social network features by T-SNE [30]. The nodes are picked from the test set covering three labels. Fig. 7a represents the node vectors from X_{MA} and Fig. 7b represents the vectors from X_{MM_AA}. Different colors denote different labels.

The MA network can distinguish services belonging to different categories well. For example, most of the purple nodes are with very short distances and there forms a cluster. Note that the big green square is *PixelPipe* and the big blue circle is *Smugmug*, and their locations are very close. They and the other cyan nodes around them denote the service social network of Fig. 2. This proves that MA makes contributions to label recommendation by projecting services of the same category directly into neighboring locations. However, it seems MM-AA cannot separate the nodes with different labels. Considering the constructions of MM and AA networks which are both based on co-occurrence of services, mashups are independent of APIs in the MM-AA. Therefore, MM-AA cannot make precise label recommendations by itself. Besides, co-occurrence implies that a service might belong to cross-category, so MM-AA tends to improve top-5 accuracy rather than top-1 accuracy. As the MA network is trained on a small dataset, it can be easily affected by the MM-AA, and some structural information is corrupted. This is the main reason that the top-1 accuracy of ALL-Cat is unsatisfying.

2) Effect of Service Social Network in SRaSLR: Fig. 8 demonstrates the visualization of Bert-Baseline's output, in which the transparency degree of nodes is used to indicate the description length of the corresponding services. A node is more transparent if the description is longer (this figure cannot be used to evaluate the performance of Bert-Baseline). Note that there is a cluster at the top-right position, in which nodes are nearly opaque (i.e., they have long descriptions), and short and long descriptions are separated significantly.

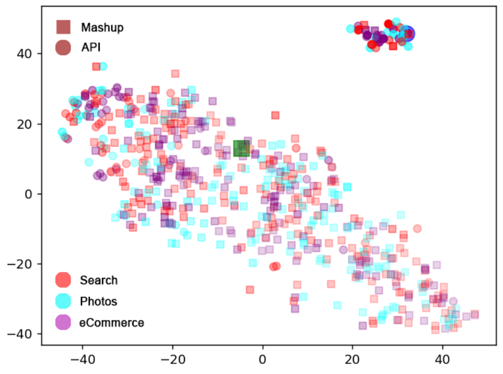


Fig. 8: Visualization of text features

TABLE III: Top-5 Accuracy on different text length

	All	Services with short description	Services with long description
Bert-Baseline	0.605	0.518	0.713
MA	0.538	**0.571**	0.539
ALL-Cat	0.544	<u>0.554</u>	0.530
(MA)	<u>0.625</u>	<u>0.554</u>	<u>0.696</u>
(ALL-Cat)	**0.636**	**0.571**	**0.744**

We divide some of the samples in the test set into two groups, short descriptions (no more than 15 words) and long descriptions (longer than 80 words), and then rerun the same test process on the new test sets. Table III shows the results. Bert-Baseline works well with long descriptions due to its powerful feature extraction capability, and it reaches 71% top-

5 accuracy. But when dealing with short descriptions, the accuracy goes down about 20%. This is mainly because a short description is unlikely to contain enough information related to the label. Instead, whatever the description lengths are, performances of the service social network based approaches are similar and better than Bert, especially when the descriptions are short. This is because the structural information is fully utilized. Moreover, SRaSLR(MA) and SRaSLR(ALL-Cat) outperform the others and achieve the highest top-5 accuracy, no matter how long the descriptions are. It is clear that networks are able to co-work with Bert and they complement each other. Combining them together, our model takes full advantage of the valuable information provided by both sides.

VII. Conclusion and Future Works

In this paper, we propose a novel social relation aware service label recommendation model SRaSLR which combines features in service descriptions and in service social networks for service label recommendation. Extensive experimental results show that SRaSLR outperforms the baselines. Further experiments and analysis demonstrate that service social networks make significant contributions to the label recommendation task, especially when service description is short and lacks enough information.

Future work is mainly focused on two perspectives: (1) to improve the graph embedding method by importing deep learning methods into the service social network encoding; (2) to make a further study on multiple fusion methods for better representing a service vector. This would push forward more valuable research in this domain.

Acknowledgement

Research in this paper is supported by the National Key Research and Development Program of China (2018YFB1402500) and the National Science Foundation of China (61832004, 61772155, 61832014, 61802089).

References

[1] P. Hasselmeyer, "On service discovery process types," in *International Conference on Service-Oriented Computing*. Springer, 2005, pp. 144–156.

[2] W. Lo, J. Yin, and Z. Wu, "Accelerated sparse learning on tag annotation for web service discovery," in *2015 IEEE International Conference on Web Services*. IEEE, 2015, pp. 265–272.

[3] X. Liu, S. Agarwal, C. Ding, and Q. Yu, "An lda-svm active learning framework for web service classification," in *2016 IEEE International Conference on Web Services*. IEEE, 2016, pp. 49–56.

[4] Y. Yang, W. Ke, W. Wang, and Y. Zhao, "Deep learning for web services classification," in *2019 IEEE International Conference on Web Services*. IEEE, 2019, pp. 440–442.

[5] Y. Yang, N. Qamar, P. Liu, K. Grolinger, W. Wang, Z. Li, and Z. Liao, "Servenet: A deep neural network for web services classification," in *2020 IEEE International Conference on Web Services*. IEEE, 2020, pp. 168–175.

[6] O. Adeleye, J. Yu, S. Yongchareon, and Y. Han, "Constructing and evaluating an evolving web-api network for service discovery," in *International Conference on Service-Oriented Computing*. Springer, 2018, pp. 603–617.

[7] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual Review of Sociology*, vol. 27, no. 1, pp. 415–444, 2001.

[8] H. Cai, V. W. Zheng, and K. C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1616–1637, 2018.

[9] I. Katakis, G. Meditskos, G. Tsoumakas, N. Bassiliades *et al.*, "On the combination of textual and semantic descriptions for automated semantic web service classification," in *IFIP International Conference on Artificial Intelligence Applications and Innovations*. Springer, 2009, pp. 95–104.

[10] X. Wang, F. Chen, and M. Li, "Web service classification approach with an integrated similarity measure," in *23rd International Conference on Industrial Engineering and Engineering Management*. Springer, 2017, pp. 251–255.

[11] E. Mavridou, G. Hassapis, D. D. Kehagias, and D. Tzovaras, "Semantic categorization of web services based on feature space transformation," in *2012 16th Panhellenic Conference on Informatics*. IEEE, 2012, pp. 162–167.

[12] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, "Deep learning based text classification: A comprehensive review," *CoRR*, vol. abs/2004.03705, 2020. [Online]. Available: https://arxiv.org/abs/2004.03705

[13] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," in *25th International Joint Conference on Artificial Intelligence*. IJCAI/AAAI Press, 2016, pp. 2873–2879.

[14] Y. Cao, J. Liu, B. Cao, M. Shi, Y. Wen, and Z. Peng, "Web services classification with topical attention based bi-lstm," in *International Conference on Collaborative Computing: Networking, Applications and Worksharing*. Springer, 2019, pp. 394–407.

[15] H. Ye, B. Cao, Z. Peng, T. Chen, Y. Wen, and J. Liu, "Web services classification based on wide & bi-lstm model," *IEEE Access*, vol. 7, pp. 43 697–43 706, 2019.

[16] R. Johnson and T. Zhang, "Supervised and semi-supervised text categorization using lstm for region embeddings," in *International Conference on Machine Learning*. PMLR, 2016, pp. 526–534.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *31st International Conference on Neural Information Processing Systems*, 2017, pp. 6000–6010.

[18] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. ACL, 2019, pp. 4171–4186.

[19] K. Punitha, "A novel mixed wide and pso-bi-lstm-cnn model for the effective web services classification," *Webology*, vol. 17, no. 2, pp. 218–237, 2020.

[20] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[21] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2013.

[22] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *20th ACM International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 701–710.

[23] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *22nd ACM International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 855–864.

[24] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *22nd ACM International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1225–1234.

[25] D. Benslimane, S. Dustdar, and A. Sheth, "Services mashups: The new generation of web applications," *IEEE Internet Computing*, vol. 12, no. 5, pp. 13–15, 2008.

[26] A. Pal, M. Selvakumar, and M. Sankarasubbu, "Magnet: Multi-label text classification using attention-based graph neural network," in *The International Conference on Agents and Artificial Intelligence*, 2020, pp. 494–505.

[27] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger,

M. Drame, Q. Lhoest, and A. M. Rush, “Transformers: State-of-the-art natural language processing,” in *2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. ACL, 2020, pp. 38–45.

[28] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[29] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *3rd International Conference on Learning Representations*, 2015.

[30] L. Van der Maaten and G. Hinton, “Visualizing data using t-sne,” *Journal of Machine Learning Research*, vol. 9, no. 11, pp. 2579–2605, 2008.