World Wide Web (2020) 23:2005–2022
https://doi.org/10.1007/s11280-020-00782-2

Geographical address representation learning for address matching

Shuangli Shan[1] · Zhixu Li[1] · Qiang Yang[2] · An Liu[1] · Lei Zhao[1] · Guanfeng Liu[3] · Zhigang Chen[4,5]

Received: 1 November 2019 / Revised: 15 December 2019 / Accepted: 2 January 2020 /
Published online: 28 February 2020

Abstract
Address matching is a crucial task in various location-based businesses like take-out services and express delivery, which aims at identifying addresses referring to the same location in address databases. It is a challenging one due to various possible ways to express the address of a location, especially in Chinese. Traditional address matching approaches relying on string similarities and learning matching rules to identify addresses referring to the same location, could hardly solve the cases with redundant, incomplete or unusual expression of addresses. In this paper, to learn the geographical semantic representations for address strings, we novelly propose to get rich contexts for addresses from the Web through Web search engines, which could strongly enrich the semantic meaning of addresses that could be learned. Apart from that, we propose a two-stage geographical address representation learning model for address matching. In the first stage, we propose to use an encode-decoder architecture to learn the semantic vector representation for each address string where an up-sampling and sub-sampling strategy is applied to solve the problem of address redundancy and incompleteness. The attention mechanism is also applied to the model to highlight important features of addresses in their semantic representations. And in the second stage, we construct a single large graph from the corpus, which contains address elements and addresses as nodes, and the edges between nodes are built by word co-occurrence information to learn embedding representations for all the nodes on the graph. Our empirical study conducted on two real-world address datasets demonstrates that our approach greatly improves both precision (up to 8%) and recall (up to 12%) of the state-of-the-art existing methods.

Keywords Address matching · LSTM · Attention Mechanism · GCN

This article belongs to the Topical Collection: *Special Issue on Web and Big Data 2019*
Guest Editors: Jie Shao, Man Lung Yiu, and Toyoda Masashi

✉ Zhixu Li
zhixuli@suda.edu.cn

Extended author information available on the last page of the article

1 Introduction

Nowadays, address localization becomes a core function in various location-based businesses like take-out services and express delivery. However, due to various address expression standards as well as typing errors in human inputs, address parsing and understanding becomes a big obstacle in address localization. The problem becomes more serious for addresses in Chinese, given more complex and diverse expression such as the examples listed in Table 1.

For better address parsing and understanding, **address matching** task has been studied [7], aiming at identifying addresses referring to the same location across different address databases. Traditional methods for address matching or standardization rely on approximate string matching metrics such as edit distance, which are not robust to deal with various kinds of expressions. A more widely-used way for address matching is to build a decision tree consisting of learned matching rules, where each rule corresponds to a path from the root node to a leaf node on the tree. Based on this so-called *address matching tree*, the similarities between two addresses could be computed. To deal with the fuzzy expressions of the addresses, some work proposes to use the forward maximum matching algorithm to segment the address to match entries in a standard address dictionary, referring to the learned matching rules [4]. Some other work also combines edit distance with space vector model to calculate the similarity between strings [16]. They measure the dissimilarity caused by the differences in the characters of address strings using edit distance, and calculate the dissimilarity caused by differences in the address terms using TF-IDF term weighting. The final result is obtained by weighting the two dissimilarities.

However, the existing methods relying on string similarities and rule matching can be easily influenced by the diverse expressions including redundant, incomplete or unusual expression of addresses. For example in Table 1, where all the five addresses refer to the same location. Assume we have a matching rule that `CITY+STREET+STREET_NO.+POI_NAME` → `ADDRESS`, which means, addresses having the same city name, street name, street No., and POI name should refer to the same POI location. Based on this rule, we may identify that *addr1* and *addr2* refer to the same location, but we could hardly judge that *addr3*, *addr4* and *addr5* also refer to the same location with *addr1*, given that *addr3* is represented by the orientation information of the adjacent address, *addr4* lacks the information of poi, and *addr5* uses the road intersection to describe the address.

Table 1 Five different address expressions for the same location

ID	Address
add1	RT-MART, Danling road No.18, Haidian district, Beijing (北京市海淀区丹棱路18号乐天玛特)
add2	RT-MART, Chuangfu Building, Danling road No.18, Beijing (北京市丹棱路18号创富大厦乐天玛特)
add3	RT-MART, Seismological Bureau of Beijing south 50 meters, Haidian district, Beijing (北京市海淀区地震局向南50m乐天玛特)
add4	Chuangfu Building 1106, Beijing (北京市创富大厦1106)
add5	RT-MART, crossroad of Caihefang Road and Danling Road, Haidian district, Beijing (北京市海淀区彩和坊路和丹棱路交叉口乐天玛特)

To address the weaknesses of the existing approaches, we turn to encode addresses into semantic space and find alignment according to its embedding similarities. It is well-known that deep learning techniques have achieved great success on word embedding [21], entity embedding [11] and sentence embedding [14, 26] in recent year. But the task of address embedding is untouched. Different from the general representation learning models which are for general domains, address representation is for a specific domain. Take the addresses in Table 1 for example, the general sentence representation models would not support that all the five addresses are close in semantic. But as a domain representation model, our deep semantic representation model should learn that "Chuangfu Building" and "RT-MART", "Danling road" and "Caihefang Road" are semantically close to each other in geography.

To learn the **geographical semantic representations** for address strings, we first need to collect a large corpus of the address localization domain for model learning. In this paper, we propose to collect the address corpus from the Web with the help of Web search engines, given the assumption that all the addresses should be mentioned, wholely or partially, at somewhere on the Web. Therefore, we may learn contexts for all these addresses and then learn deep semantic representations for them from the contexts. Based on the collected address corpus from the Web, we propose a two-stage geographical address representation learning model for address matching. In the first stage, we propose to use an encode-decoder architecture with two Long Short Term Memory (LSTM) networks [10], where one is regarded as encoder and the other as decoder, to learn the semantic vector representation for each address string.

Due to the problems of address redundancy and addresses incomplete, we up-sample and sub-sample the address during the training process, then take the sampled address as the input of the encoder and use the original address as the output of the decoder to improve the robustness of the model. We also apply an attention layer between encoder and decoder to enable the semantic vectors to represent richer semantic information by assigning higher weights to more important features. In the second stage, we construct a single large graph from the corpus, which contains address elements and addresses as nodes, and the edges between nodes are built by word co-occurrence information. We then model the graph with GCN [27] to learn representations for all the nodes on the graph. Based on the graph embedding results, the address matching results could be greatly improved even with a small proportion of labeled data.

We summarize our contributions in this paper as follows:

- We novelly propose to measure the semantic similarity between addresses for address matching based on deep semantic address representation.
- We novelly propose to get rich contexts for addresses from the Web through Web search engines, which could strongly enrich the semantic meaning of addresses that could be learned.
- Based on the collected address corpus from the Web, we propose a two-step geographical address representation learning model for address matching.

We perform experiments on two real-world datasets, and the empirical results demonstrate that our proposed model works much better than the state-of-the-art methods on both precision (up to 8%) and recall (up to 12%). ***Roadmap.*** The rest of the paper is organized as follows: we give a formal definition to address matching problem in in Sec. 2. After presenting a compound framework for address matching in Sec. 3, we then present our geographical address representation learning model in Sec. 4. After reporting our experiments in Sec. 5, we cover the related work in Sec. 6. We conclude in Sec. 7.

2 Problem definition

Address localization is a core task in various location-based businesses. Depending on whether there is a standard address database, address localization can be divided into address standardization task and address matching task. In this paper, we study on the case without a standard address database, i.e., address matching task, which is formally defined below:

> **Definition (Address Matching.)** Given a set of addresses $=\{add_1, add_2, \ldots, add_n\}$, the goal of address matching is to find every address pair (add_i, add_j) satisfying $add_i \doteq add_j$, where $add_i \in D$, $add_j \in D$, $i \neq j$ and $\doteq$ is a comparison operator having its two operands referring to the same real-world object.

3 A compound framework for address matching

Instead of working independently, we prefer to let our deep semantic matching approach work under a compound framework, where the address matching rule tree is also adopted in the *Syntactic matching step* before going into the *semantic matching step* using deep semantic representations of addresses.

We illustrate this compound framework in Figure 1: Given a set of addresses, we first generate a number of candidate address pairs for matching based on some simple heuristic rules as introduced in [28]. For instance, only addresses in the same city and district (if any), sharing at least one word (after removing stop words) in the left part of their address strings need to be compared. For every candidate address pair, we use a basic address matching tree following [4] to decide whether the two addresses could be syntactically matched in the *Syntactic matching step*. If yes, the address pair will be put into the final matching results. Otherwise, the address pair goes into the *semantic matching step*, where vectors of the two addresses would be obtained from the learned *deep semantic address representation model*. We give more details below:

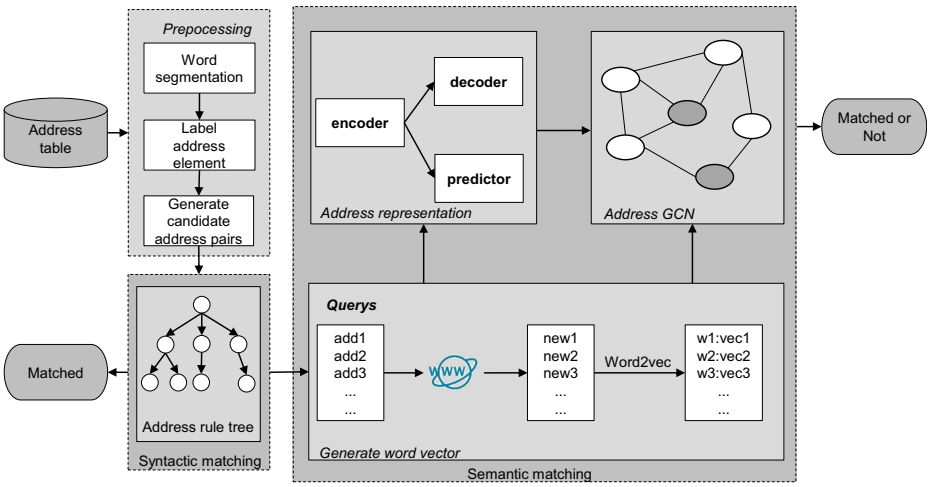


Figure 1 A compound framework with both syntactic and semantic address matching

- **Syntactic Matching.** For syntactic matching, we adopt an address matching tree containing a number of expert-defined address matching rules as the one shown in Figure 2, where every path from the root node to a leaf node corresponds to a matching rule. For every candidate address pair, we would let them traverse the whole tree in a deep-first traverse way to find if it could strictly match with a matching rule. Once successfully matched with a matching rule at its leaf node, the traverse process will be terminated.
- **Semantic Matching.** All candidate address pairs that cannot be matched in the syntactic matching step will go to the semantic matching step. For every candidate pair, we use the learned *geographical address representation model* to get the vector of the two addresses, and then compute their similarities to see if they could be matched or not according to a predefined threshold. The core module of semantic matching is how we build the *geographical address representation learning model*, which will be introduced in the following section.

4 Geographical address representation learning model

It is a nontrivial task to learn a deep semantic representation model for addresses. The general sentence representation models are built for general purposes, but address representation is a task of a specific domain.

In this paper, we propose to collect the address corpus from the Web with the help of Web search engines, based on which we then train our geographical address representation learning model in two stages. In the first stage, for every address in the training set, we crawl the latest news about it on the Web through Web search engines, and treat these news as our corpus. Next, we use the corpus to enrich the semantic information of the addresses and employ **Word2Vec** [20] to train our word vector. Based on this, we propose to use a encoder-decoder model with attention mechanism to represent sentence vectors. Different from the traditional model, we here replace one hot representation with the pre-trained word vectors as the initial vectors, which helps to capture more semantics.

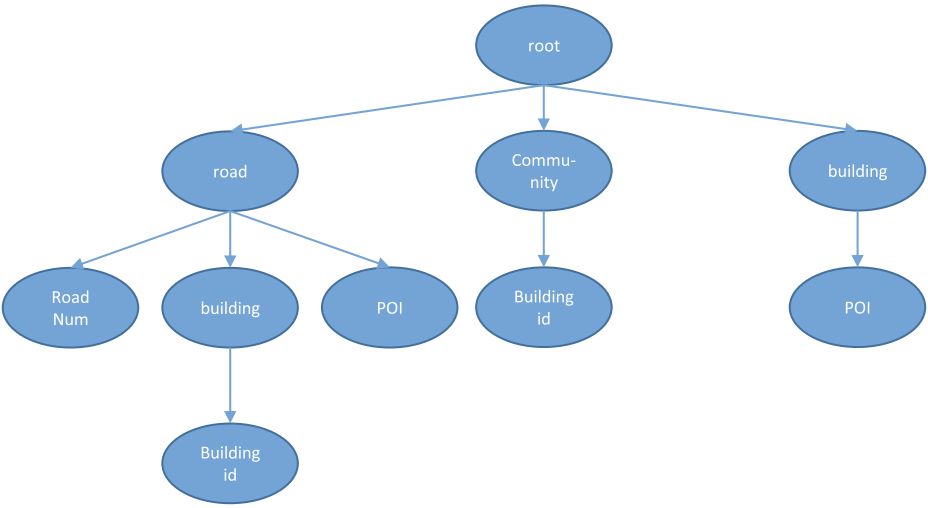


Figure 2 A basic address matching tree for syntactic address matching

In the second training stage, we construct a single large graph from the corpus, which contains address elements and addresses as nodes, and the edges between nodes are built by word co-occurrence information. We then model the graph with GCN [27] to learn representations for all the nodes on the graph. Based on the graph embedding results, the address matching results could be greatly improved even with a small proportion of labeled data.

In the rest of this section, we first introduce how we get extra contexts for a given address from the Web and perform domain word2vec training in Sec. 4.1, and then present the encoder-decoder model with attention mechanism for deep semantic address representation in the first step in Sec. 4.2. After that, we introduce the GCN-based representation learning model in Sec. 4.3.

4.1 Word embedding for the address domain

There are plenty of contexts on the Web where rich geographic semantic information can be captured. Let $A = \{e_1, e_2, e_3, ..., e_n\}$ denote a collection of elements of an address which can be gotten with word segment tool [20]. For every address element e_i, we use Web Search Engines to obtain a collection of Web page, denote as $W_i = \{p_{i1}, p_{i2}, p_{i3}, ..., p_{im}\}$. Therefore, for an address, there are a total of $n \times m$ pages. In order to remove irrelevant Web page, we use a vector space model (VSM) for representing a collection of Web pages to calculate a score for each page. Specifically, each Web page is represented by a document vector consisting of a set of weighted keywords. Keywords in each Web page are address elements. The weights of the address element in each Web page vector are determined according to the term frequency inverse document frequency (TF-IDF) model. The weight of e_i in Web page $p_j (1 < j < n \times m)$, denoted as $w(e_i, p_j)$, which can be calculated by the following formula:

$$w\left(e_i, p_j\right) = Num\left(e_i, p_j\right) \times \log \frac{m \times n + 1}{k} \tag{1}$$

where $Num(e_i, p_j)$ denotes the number of occurrences of e_i in Web page p_j and k is the number of Web pages which contain element e_i. After that, we can calculate a score for every Web page by the following equation:

$$score\left(p_j\right) = \sum_{i=1}^{n} w\left(e_i, p_j\right) \tag{2}$$

Next, we select the Web pages with high scores, e.g. larger than a predefined threshold, and then extract contexts from the Web pages. These contexts are used as the training corpora to train the word vectors related to address elements by using the Word2vec technique. A general corpus, Chinese Wikipedia [23], is also utilized to pre-train word vectors before using the Web pages for address domain. After that, for every address element $e_i \in A$, we can get its vector $\mathbf{x_i}$. Finally, we can obtain a collection of word vectors $\mathbf{W} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n}\}$ for every sentence.

4.2 Stage 1: deep address representation learning

In the second training state, we focus on learning the address representation with encoder-decoder model. Specifically, we apply LSTM on the encoder model and decoder model respectively. In addition, we utilize predictor model to predict the next sentence. In order to solve the problems of address redundancy and addresses incomplete, we up-sample and

subsample the addresses from the original training set to construct a new training set. That is, we use address matching rules to determine which address elements are unessential. After that, we remove the unessential address elements so that the addresses can be represented in different forms and we can also artificially add duplicate elements to the address. Next, we take the sampled address as the inputs of the encoder and use the original address as the output of the decoder to satisfy that the semantics of addresses with incomplete or redundancy are close. Furthermore, we apply the attention mechanism on encoder-decoder model and encoder-predictor model such at the important features can be emphasized by assigning higher weights.

(1) Encoder-decoder Model with Attention Firstly, the encoder reads the input sentence one-by-one which is an address or a sentence from Web corpus. Note that the address or the sentence here has been initialized by the word embedding shown in section 4.1, i.e. $\mathbf{W} = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$. Next, in a certain time step t of LSTM, there are three gates in a LSTM time step, input gate, forget gate and output gate respectively, denoted as i_t, f_t and o_t. They are composed of a sigmoid neural net layer and a pointwise multiplication operation and their values range from 0 to 1. Let $H = \{\mathbf{h_1}, \mathbf{h_2}, \ldots, \mathbf{h_t}, \ldots, \mathbf{h_m}\}$ be the set of hidden state for each sentence from the inputs. We use the following equations to get the hidden state $\mathbf{h_t}$ for the time step t. We first need to forget the old subject when we meet a new subject. And we decide which information we are going to throw away from the previous hidden state h_{t-1} with the Eq. (3):

$$f_t = \sigma\left(w_f * \mathbf{h_{t-1}} + w_f * \mathbf{x_t}\right) \tag{3}$$

The next step is to decide what new information we are going to store in the current state. And input gate decides which values will be updated as follows.

$$i_t = \theta(w_i * \mathbf{h_{t-1}} + w_i * \mathbf{x_t}) \tag{4}$$

Next, a tanh layer creates a vector of new candidate values $\widetilde{\mathbf{c}}_\mathbf{t}$:

$$\widetilde{\mathbf{c}}_\mathbf{t} = \tanh\left(w_c * \mathbf{h_{t-1}} + w_c * \mathbf{x_t}\right) \tag{5}$$

After the forget gate and input gate have been computed, it is time to update the old memory state $\mathbf{c_{t-1}}$ into the new state $\mathbf{c_t}$.

$$\mathbf{c_t} = f_t * \mathbf{c_{t-1}} + i_t * \widetilde{\mathbf{c}}_\mathbf{t} \tag{6}$$

where $\mathbf{c_t}$ denotes the memory state in time step t. Finally, we need to decide what we are going to output. Specifically, we run a sigmoid layer to decide which parts of the cell state are going to output.

$$o_t = \sigma(w_o * \mathbf{h_{t-1}} + w_o * \mathbf{x_t}) \tag{7}$$

Then, we put the memory state through tanh to push the values between −1 and 1 and multiply it by the output of the output gate. Therefore, we can output the parts.

$$\mathbf{h_t} = o_t * \tanh\left(\mathbf{c_t}\right) \tag{8}$$

where w_f, w_i, w_c, and w_o denote weights of each part. In this way, the entire input sentence are mapped as a fixed-length vector which is then provided as an input to the decoder model.

Then, the decoder outputs a sentence by decoding the hidden stage into the input sentence in the same manner. This process is shown in Figure 3.

Due to the hierarchical relationship of address elements, we want to determine which part of the input is most responsible for the current decoder output. Therefore, we adjust our model by adding attention mechanism. Suppose that $\mathbf{r_t}$ denotes the input word embedding at current step t and $\mathbf{h_{t-1}}$ denotes the hidden state decoder at previous time step t–1. Attention model would first mix encoder output vector $\mathbf{h_{t-1}}$ with current input word embedding $\mathbf{r_t}$.

$$v_t = f(\mathbf{h_{t-1}}, \mathbf{r_t}) \tag{9}$$

where v_t is the weight of each word vector. Generally, f is a simple feed-forward network with 1 or 2 dense layers. After that, we normalize the v_t with the following equation:

$$w_t = \frac{exp(v_t)}{\sum_{t=1}^{N} exp(v_t)} \tag{10}$$

where N is the number of words contained in a input sentence. Then, the final fixed length vector $\overrightarrow{v}$ can be calculated as follows:

$$\mathrm{v} = \sum_{t=1}^{N} w_t \mathbf{h_t} \tag{11}$$

In this way, the important words are weighted with the higher scores and then we can use information from these words to construct the fixed length vector. In addition, it can take variable length inputs instead of the fixed-length inputs.

(2) Encoder-predictor Model with Attention This model is used to predict the next sentence and obtain the geographic semantic information. Different from the encoder in the encoder-decoder model, the input of encoder in the encoder-predictor model is just a sentence in the Web corpus. And the predictor is designed to predict the following words based on the

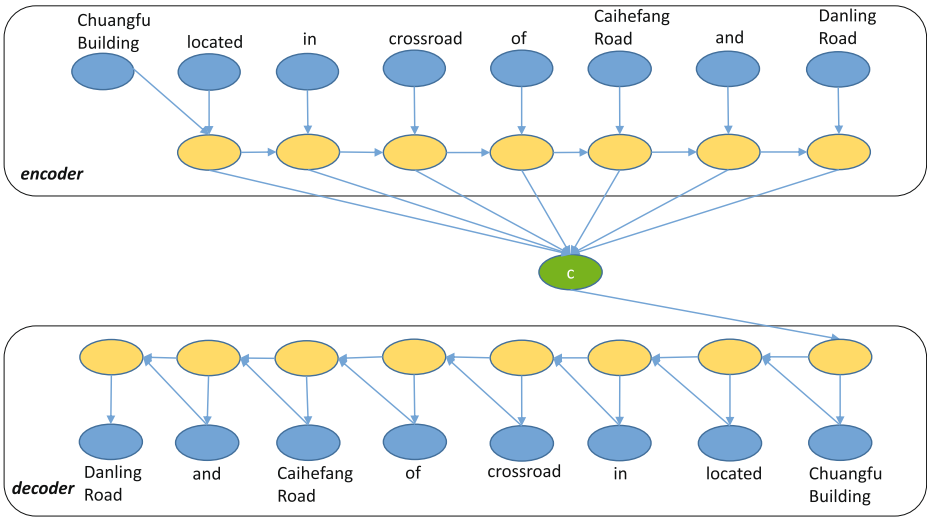


Figure 3 The illustration of encoder-decoder model

previous state. And the attention mechanism is also used in the same way. Here, we employ the conditional probability to output the sentence with the highest probability:

$$p(\mathbf{y}|\mathbf{v}) = \sum_{t=1}^{N} p(\mathbf{y_i}|\mathbf{y_1}, \ldots, \mathbf{y_{i-1}}, \mathbf{v}) \tag{12}$$

where $\mathbf{y_i}$ is defined as the output of the predictor at time step i. All the words in the sentence are sequentially generated using the LSTM until the end of the sentence symbol is generated. The training objective is to maximize the conditional probability over the corpus using stochastic gradient descent. Especially, $\mathbf{y_0}$ is defined as a zero vector. After that, we store the optimal parameters of our model for the training set. The process of the encoder-predictor model is shown in Figure 4. Finally, we use the trained model on the testing set to build vectors of addresses.

4.3 Stage 2: GCN-based address representation learning

Graph Convolutional Network (GCN) [13] is a neural network designed to work directly on graphs and leverage their structural information about graph. More formally, given a graph $G = (V, E)$, where V is a set of nodes and E is a set of edges. GCN is a special case of Graph Neural Network (GNN) which stacks two layers of specific propagation and perceptron:

$$\begin{aligned} H^{(1)} &= ReLU\Big((PX)W^{(0)}, \\ H^{(2)} &= \Big(PH^{(1)}\Big)W^{(1)} \end{aligned} \tag{13}$$

where $P = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$is the normalized symmetric adjacency matrix, D is degree matrix of G and A is its adjacency matrix, and $W^{(0)}$, $W^{(1)}$ are the weight matrixs. Furthermore, GCN can collect higher neighborhoods information by adding more GCN layers.

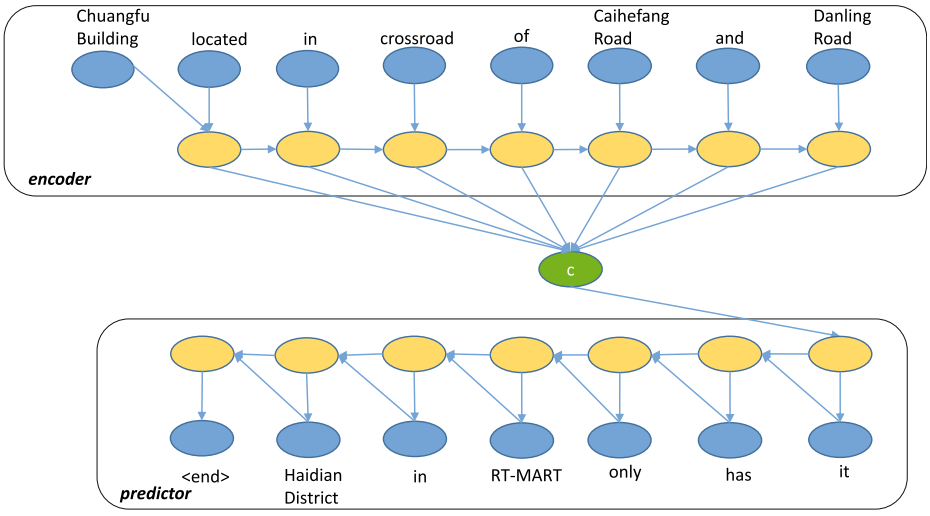


Figure 4 The illustration of encoder-predictor model

We first build a large and heterogeneous address graph containing address element nodes (like E_i) and address nodes (like A_i) where address nodes depend on address element nodes and nodes are linked by composition relationship as shown in Figure 5(left part). Note that the number of nodes connected to an address node equals to the number of unique address elements it contains. In details, for address elements nodes and the address nodes, they are initialized by Deep Semantic Address Representation Model which mentioned in Sec. 4.2, while for edges, they are formed when two address elements appear in the same address. The weight of an edge among address elements is counted based on the number of co-occurrences of them in dataset while the weight between address element nodes and address nodes are calculated with two strategies. The first one is based on the identical weight, for example, all weights are set to 1, and the second one is based on the auto-learning where attention mechanism is used to determine which neighbor address element node is more important.

After building the address graph, we use two layer GCN to update the node embeddings. For the address element nodes, our forward model then takes the simple form:

$$H^{(2)} = P^{*}ReLU\left((PX)W^{(0)}\right)W^{(1)} \tag{14}$$

where $P = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ is the same as in equation 13, and $H^{(2)}$ is the output node embeddings. For the address nodes, we use different propagation rules. The first layer network is the same as the original GCN, the parameter matrix $W^{(0)}$ transforms node X into a feature vector, and the activation function is the ReLU:

$$H^{(1)} = ReLU\left(XW^{(0)}\right) \tag{15}$$

This is followed by a layer of attention propagation layers parameterized by $\beta^{(1)} \in \mathbb{R}$, so the output vector of node i is:

$$H_i^{(2)} = W^{(2)} \sum_{j \in N(i) \cup i} P_{ij}^{(1)} H_j^{(1)} \tag{16}$$

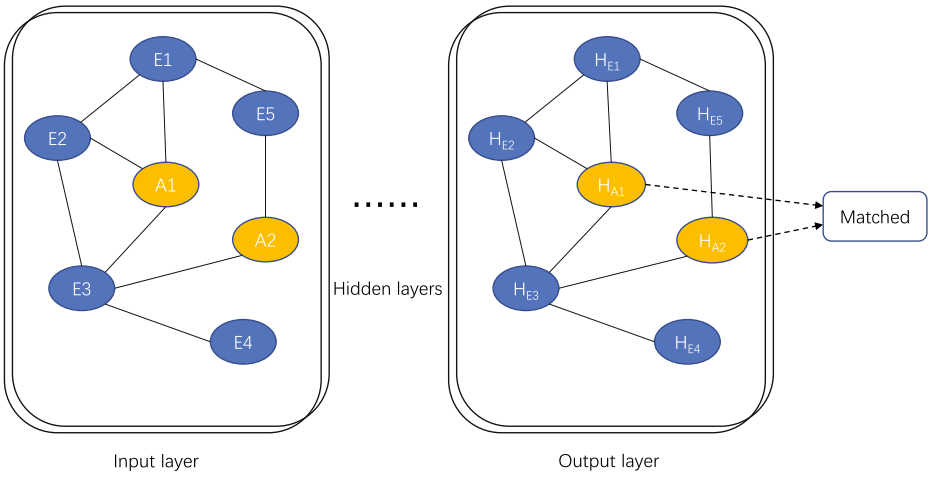


Figure 5 Geographical address representation learning process. Note that nodes starting with "E" are address element nodes, nodes starting with "A" are address nodes, and nodes starting with "H" are the output vectors

$$P_{ij}^{(1)} = \frac{e^{\beta^{(1)} cos\left(H_i^{(1)}, H_j^{(1)}\right)}}{\sum_{j \in N(i) \cup i} e^{\beta^{(1)} cos\left(H_i^{(1)}, H_j^{(1)}\right)}}, j \in N(i) \cup i \tag{17}$$

where $N(i)$ denotes the neighborhood of node i and $P_{ij}^{(1)}$ denotes the attention from node j to node i. The cosine distance between node i and node j in the hidden state of layer is to capture association degree between node i and node j. The attention mechanism is to select the neighbor address element nodes that are more relevant to the central address node. Finally, we calculate the probability that the two addresses are matched as follows:

$$Z_{i,j} = sigmod\left(cosine\left(H_i^{(2)}, H_j^{(2)}\right)\right) \tag{18}$$

The weights $W^{(0)}$, $W^{(1)}$ and $\beta^{(1)}$ are trained to minimize the cross entropy loss over all labeled examples L, labels are denoted by $y_{i,j}$:

$$\pounds = -\sum_{i,j \in L} y_{i,j} ln(Z_{i,j}) - \sum_{i,j \in L} (1 - y_{i,j}) ln(1 - Z_{i,j})$$

5 Experiments

This section presents our experimental results. We run the experiments on the a server with a GTX-1080ti gpu, four-core Intel Core i7 processor, 16GB memory, running centos. All the models are implemented using Tensorflow [1]. The size of word embedding dimensionality is set as 300 [21] and the learning rate for DeepAM models is 0.002. For Address GCN, we train the model for a maximum of 200 epochs with a learning rate of 0.01 and set window size as 10 [17].

5.1 Datasets and metrics

For evaluation, we compare our proposed method with several existing state-of-the-art methods on two real-world datasets below:

- **POI.** We collect addresses of SuZhou city from a Chinese POI website[1] where every POI has one corresponding address. This dataset contains 200 k pieces of POI addresses, and 6 different districts. The expressions of addresses in this dataset are diverse and they also have some redundant or incomplete information.
- **Company.** We also crawl addresses of SuZhou city on two food review websites[2,3] and one company information query website.[4] This database contains 10 k company addresses, and it has two different districts. The problems of the redundant, incomplete information and diverse expressions for addresses in this dataset becomes even worse.

[1] www.poi86.com
[2] www.dianping.com
[3] www.meituan.com
[4] www.qichacha.com

Metrics. We basically use three metrics to evaluate the effectiveness of the methods: **Precision:** the percentage of correctly matched pairs among all address pairs. **Recall:** the percentage of correctly matched pairs among all address pairs that should be matched. **F1 Score:** a combination of precision and recall, which is calculated by $F1 = \frac{2*precision*recall}{precision+recall}$.

5.2 Methods for comparison

In this section, we introduce our proposed method *Address GCN* with the existing methods including *String-Based* method, *Dictionary-Based* method, and *Address Matching Tree* method.

- The *String-Based* method combines the vector space model and edit distance to calculate the address similarity.
- The *Dictionary-Based* method uses a place-name dictionary to parse the address elements such that it can reduce the effect of place-name ambiguity on address matching.
- The *Address Matching Tree* (AMT for short) method builds up a rule-based matching tree and applies it to do address matching by transforming address matching tree into a set the matching rules.
- The *DeepAM* method utilizes Web Search Engines to get extra contexts for the address as the corpus and uses an encoder-decoder architecture with two LSTM networks to get the address representation. The vectors of address pairs are built based on the model to compute the similarities.
- The Address GCN method build a large and heterogeneous address graph from all corpus and fed it into a two-layers graph convolution network to obtain more geographic information.

5.3 Comparisons with previous methods

In this section, we compare the effectiveness of our proposed address matching method with the mentioned methods above.

As is shown in Figure 6, the String-Based method has the lowest F1 scores, because it just considers the address similarity based on string which cannot tackle the problem of diversity

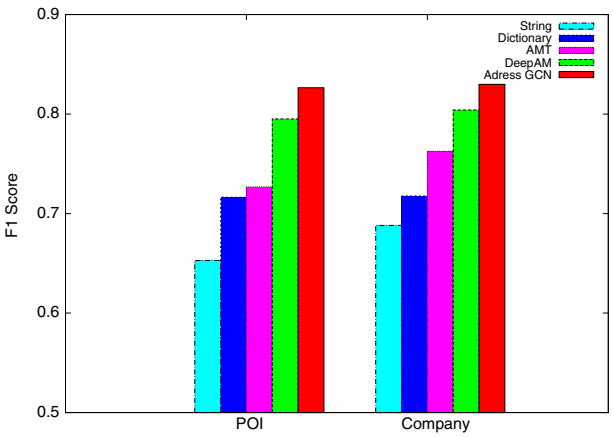


Figure 6 Comparing with previous methods on F1 score

well. The accuracy of the Dictionary-based method is lower than the AMT method since it greatly relies on the quality of place-name dictionary. DeepAM method performs better in that it extracts the geographic information from the Web and use sentence representation model to learn the deep semantic information. Our proposed Address GCN method have the highest precision and recall because it can obtain spatial structure information of the address.

In addition, we compare the Precision and Recall of these methods on the two datasets. As is listed in Table 2, the Address GCN method also reaches the highest precision and recall among all methods, which is followed by the DeepAM method with the second highest values. The precision and recall of AMT method is better than Dictionary-based method and String-based method is the worst one. We can see that address GCN in our model works well and improves both the precision and recall. The reason why Address GCN works well is analyzed as follows: GCN model which is a special form of Laplacian smoothing, can compute the new features of a node as the weighted average of itself and its second order neighbors [17].

5.4 Evaluation of the size of labeled address pairs

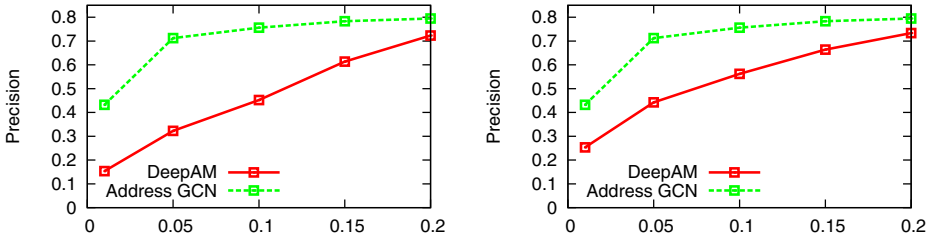
In order to evaluate the effect of the size of the labeled Address pairs, we tested our proposed models with different proportions of the training data. Figure 7 shows matching precision with 1%, 5%, 10%, 15%, 20% of POI and Company training set. We note that Address GCN can achieve higher test precision with little labeled Address pairs. For instance, Address GCN achieves a precision of 80% with only 20% training address pairs which are higher than some baselinemodels with the full training documents. These results are similar to results in [13] where GCN can perform quite well with low label rate, which again suggests that GCN can propagate labeled information to entire graph well.

5.5 Evaluation of quality of Web contexts

In this section, we evaluate the influence on the matching precision for the quality of Web contexts. As shown in Figure 8 (a), “Percentage of Web contexts” represents Web contexts used for training as a percentage of total text, and the precision of DeepAM method first rises when the percentage of Web contexts increases, then it reaches the highest points at 0.4. After that, it has a little drop. This indicates that the model learns richer semantic information when it has lager Web contexts in the address domain. But when this figure continues to go up, it shows a decreasing trend since the irrelevant Web content has a negative effect. We can also see from Figure 8 (b), for the Company dataset, the experimental results show a similar

Table 2 Comparing with baselines on precision and recall on two datasets

Methods	POI		Company	
	Precision	Recall	Precision	Recall
String	0.6854	0.6232	0.7015	0.6753
Dictionary	0.7504	0.6854	0.7432	0.6939
AMT	0.7752	0.6843	0.7945	0.7332
DeepAM	0.8249	0.7674	0.8372	0.7736
Adress GCN	.8513	.8032	.8659	0.7968



(a) POI Dataset (b) Company Dataset

Figure 7 Effect of the size of labeled address pairs

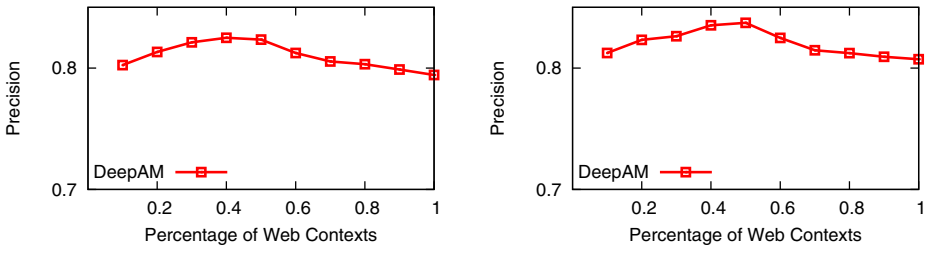
variation tendency for the precision. Therefore, Web contexts have a good effect instead of bad effect on address matching, though Web contexts have some noise.

Lastly, we list some examples of Web contexts we get for some example addresses in Table 3. As can be observed, theses Web contexts contain rich semantic information about addresses, which is beneficial to get more accurate sentence representation, such as the location, adjacent objects and so forth.

6 Related work

Given the complexity and diversity of address expressions, it is difficult to form a unified standard address model and geocoding standard, which results in the hardness of address data sharing among different government sectors and industries. To tackle the problem, a wide variety of methods have been proposed for address matching and address standardization.

A widely-used way for address matching is to build a decision tree consisting of learned matching rules, where each rule corresponds to a path from the root node to a leaf node on the tree [34]. Based on this so-called *address matching tree*, the similarities between two addresses could be computed. Kang et al. propose an address matching tree model based on the analysis of the spatial constraint relationship of address elements [19]. This method has a higher address matching rate, but it needs to establish a variety of complex address models and needs to determine the spatial constraint between the address elements. An improved hash structure-based maximum inverse matching algorithm is proposed [6]. This method can make full use of the hash function to improve the retrieval efficiency, however, it also has the disadvantage of being sensitive to the address hierarchy.



(a) POI Dataset (b) Company Dataset

Figure 8 Effect of the quality of Web contexts

Table 3 An example of Web contexts

Address element	Context
Chuangfu Building	Chuangfu Building is located at No. 18 Danling Street, Haidian District. The total construction area of Chuangfu Building is 30,408 square meters, including 17 floors above ground and 3 underground floors. Chuangfu Building is adjacent to Sihuan, Zhongguancun Street and Caihefang Road, enjoying the convenience of the city. The supporting facilities around Chuangfu Building are complete: close to the Municipal Industry and Commerce Bureau, the Earthquake Administration, major ministries and political organizations. (创富大厦由北京海淀科技园建设股份有限公司开发的,创富大厦项目总建筑面积30408平方米，其中地上17 层，地下3层。创富大厦毗邻四环、中关村大街和彩和坊路，尽享都市便捷。创富大厦周边配套设施齐备：紧邻市工商局，地震局和各大部委及政要机构.)
	Chuangfu Building is located in the central location of modern science and technology development. It is located in the southwestern corner of Zhongguancun West District. It is located at the crossroad of Danling Road and Caihefang Road, west of Suzhou Street and south of Haidian South Road. (创富大厦位于现代科技发展中心性区位，中关村西区内的西南金角位置，雄踞于丹棱路与彩和坊路交叉路口,西邻苏州街,南依海淀南路.)
	.....
Wangfujing Street	Wangfujing Commercial Street has a large number of large shopping malls from south to north. There are Dongfang Xintiandi in the southeast of Wangfujing Commercial Street, Wangfujing Department Store in the southwest, and New Dong'an Market in the northeast, forming the Wangfujing Commercial Street. (王府井商业街有南到北的大型商场很多，有位于王府井商业街东南的东方新天地，西南的王府井百货商场，还有在东北的新东安市场，组成了王府井商业街.)
	Wangfujing street, located in the north side of east Changan street in the center of Beijing, is a famous commercial district with a long history for hundreds of years. (王府井步行街，位于北京市中心的东长安街北侧，是具有数百年悠久历史的著名商业区)
	.....

Some recent efforts propose to do address matching based on the semantic analysis to the addresses. They get the semantic vectors of address expressions and then apply them to compute the similarity to decide the matching results. Song et al. apply the Chinese word segmentation and semantic inference (HMM model) in natural language processing to deal with unstructured Chinese addresses [25]. The disadvantage of this approach has a strong dependence on the HMM model training set, and it requires a large number of addresses to train the model. Pu et al. convert the task of address matching to the comparison of semantic distance on Bayesian network [22]. However, it can not accurately identify multiple ambiguous addresses.

The process of translating manually written addresses into a certain digital format is known as address standardization. There are also some researches on Address Standardization [8, 12, 15, 31]. A method based on trie-tree and finite state machine is proposed in [18] which focuses on the problem of inaccurate word segmentation. They use trie-tree to realize chinese word segmentation and apply finite state machine to match each layer of address. However, address hierarchy is too complex resulting in low matching efficiency. In addition, Wang et al. propose a standardization method based on the Euclidean distance between the address to be processed

and the address in the standard library, but it only works well on some specific data sets [30]. Furthermore, Shikhar et al. use a fully connected neural network with a single hidden layer to tackle the syntactic and semantic challenges in a large number of addresses [24]. However, it needs a large of labeled training samples.

Graph Neural Networks has received growing attention recently [29, 32]. A number of neural network models like CNN that apply to regular grid structure to work on arbitrarily structured graphs [3, 9, 33]. In previous work, Kipf and Welling presented a simplified graph neural network model, called graph convolutional networks (GCN), which achieved state-of-the-art classification results on a number of datasets. GCN wasalso used to solve several NLP tasks such as semantic role labeling [5] and machine translation [2], where GCN is used to encode syntactic information.

7 Conclusions and future work

Address matching is a crucial task in various location-based businesses like take-out services and express delivery. In this paper, we propose to use an encoder-decoder architecture with two LSTM networks to learn the semantic vector representation for an address string. Then we build an address graph and use GCN to update representation of the address node. We also propose to get rich contexts for addresses from the Web which could strongly enrich the semantic meaning of addresses that could be learned. Our experiments conducted on two real-world datasets demonstrate that our proposed model works much better than the state-of-the-art methods on both precision (up to 8%) and recall (up to 12%). In our future study, we consider to involve more geographical information into the deep semantic address embedding.

Acknowledgments This research is partially supported by Natural Science Foundation of Jiangsu Province (No. BK20191420), National Natural Science Foundation of China (Grant No. 61632016, 61572336, 61572335, 61772356), Natural Science Research Project of Jiangsu Higher Education Institution (No. 17KJA520003, 18KJA520010), and the Open Program of Neusoft Corporation (No. SKLSAOP1801).

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org (2015)
2. Bastings, J., Titov, I., Aziz, W., Marcheggiani, D., Sima'an, m, K.: Graph convolutional encoders for syntax-aware neural machine translation. arXiv preprint arXiv:1704.04675, (2017)
3. Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203 (2013)
4. Cheng, C.-x., Yu, B.: A rule-based segmenting and matching method for fuzzy chinese addresses [j]. Geography and Geo-Information Science. **3**, 007 (2011)
5. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In Advances in neural information processing systems. 3844–3852 (2016)
6. Ding, Z.-g., Zhang, Z., Li, J.: Improvement on reverse directional maximum matching method based on hash structure for chinese word segmentation. Computer Engineering and Design. **29**(12), 3208–3211 (2008)

7. Drummond, W.J.: Address matching: Gis technology for mapping human activity patterns. J. Am. Plan. Assoc. **61**(2), 240–251 (1995)
8. Guo, H., Zhu, H., Guo, Z., Zhang, X.X., Su, Z.: Address standardization with latent semantic association. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 1155–1164. ACM, (2009)
9. Henaff, M., Bruna, J., LeCun, Y.: Deep convolutional networks on graph-structured data. arXiv preprint arXiv:1506.05163, (2015)
10. Hochreiter, S., Schmidhuber, J.: Lstm can solve hard long time lag problems. In Advances in neural information processing systems. 473–479 (1997)
11. Hu, Z., Huang, P., Deng, Y., Gao, Y., Xing, E.: Entity hierarchy embedding. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 1292–1300, (2015)
12. Kaleem, A.B.D.U.L., Ghori, K.M., Khanzada, Z., Malik, M.N.: Address standardization using supervised machine learning. Interpretation. **1**(2), 10 (2011)
13. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
14. Kiros, R., Zhu, Y., Salakhutdinov, R.R., Zemel, R., Urtasun, R., Torralba, A., Fidler, S.: Skip-thought vectors. In Adv. Neural Inf. Proces. Syst. 3294–3302 (2015)
15. Kothari, G., Faruquie, T.A., Subramaniam, L.V., Prasad, K.H., Mohania, M.K.. Transfer of supervision for improved address standardization. In Pattern Recognition (ICPR), 20th International Conference on, pages 2178–2181. IEEE, (2010)
16. Li, D., Wang, S., Mei, Z.: Approximate address matching. In 2010 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, pages 264–269. IEEE, (2010)
17. Li, Q., Han, Z., Wu, X.-M.: Deeper insights into graph convolutional networks for semi-supervised learning. In Thirty-Second AAAI Conference on Artificial Intelligence, (2018)
18. Luo, M., Huang, H.: New method of chinese address standardization based on finite state machine theory. Application Research of Computers, (2016)
19. Mengjun, K., Qingyun, D., Mingjun, W.: A new method of chinese address extraction based on address tree model. Acta Geodaetica et Cartographica Sinica. **44**(1), 99–107 (2015)
20. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems, pages 3111–3119 (2013)
21. Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pages 1532–1543 (2014)
22. Pu-le, X., Yang, W., Ya-kun, H., Shao-fen, H., Chuan-xin, Z., Fu-long, C.: Chinese place-name address matching method based on large data analysis and bayesian decision. Computer Science. **9**, 050 (2017)
23. Qiu, Y., Li, H., Shen, L., Jiang, Y., Hu. R., Yang, L.: Revisiting correlations between intrinsic and extrinsic evaluations of word embeddings. In Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data, pages 209–221. Springer (2018)
24. Sharma, S., Ratti, R., Arora, I., Solanki, A., Bhatt, G.: Automated parsing of geographical addresses: A multilayer feedforward neural network based approach. In Semantic Computing (ICSC), 2018 IEEE 12th International Conference on, pages 123–130. IEEE (2018)
25. Song, Z.: Address matching algorithm based on chinese natural language understanding [j]. J. Remote Sens. **17**(4), 788–801 (2013)
26. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In Advances in neural information processing systems, pages 3104–3112 (2014)
27. Thekumparampil, K.K., Wang, C., Oh, S., Li, L.-J.: Attention-based graph neural network for semi-supervised learning. arXiv preprint arXiv:1803.03735 (2018)
28. Tian, Q., Ren, F., Hu, T., Liu, J., Li, R., Qingyun, D.: Using an optimized chinese address matching method to develop a geocoding service: A case study of shenzhen, China. ISPRS International Journal of Geo-Information. **5**(5), 65 (2016)
29. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P.S.: A comprehensive survey on graph neural networks. arXiv preprint arXiv:1901.00596 (2019)
30. Yong, W., Jiping, L.I.U., Qingsheng, G.U.O., An, L.U.O.: The standardization method of address information for pois from internet based on positional relation. Acta Geodaetica et Cartographica Sinica. **45**(5), 623–630 (2016)
31. Zhu, X., Gan, J., Lu, G., Li, J., and Zhang, S.: Spectral clustering via half-quadratic optimization. World Wide Web, https://doi.org/10.1007/s11280-019-00731-8. (2019)

32. Zhu, X., Li, X., Zhang, S., Xu, Z., Yu, L., Wang, C.: Graph pca hashing for similarity search. IEEE Transactions on Multimedia. **19**(9), 2033–2044 (2017)
33. Zhu, X., Zhang, S., Hu, R., He, W., Lei, C., Zhu, P.: One-step multi-view spectral clustering. IEEE Trans. Knowl. Data Eng. **31**(10), 2022–2034 (2019)
34. Zhu, X., Zhang, S., Li, Y., Zhang, J., Yang, L., Fang, Y.: Low-rank sparse subspace for spectral clustering. IEEE Trans. Knowl. Data Eng. **31**(8), 1532–1543 (2019)

Affiliations

Shuangli Shan[1] · Zhixu Li[1] · Qiang Yang[2] · An Liu[1] · Lei Zhao[1] · Guanfeng Liu[3] · Zhigang Chen[4,5]

Shuangli Shan
shanshuangli1@gmail.com

Qiang Yang
qiang.yang@kaust.edu.sa

An Liu
anliu@suda.edu.cn

Lei Zhao
zhaol@suda.edu.cn

Guanfeng Liu
guanfeng.liu@mq.edu.au

Zhigang Chen
zgchen@iytek.com

[1] Institute of Artificial Intelligence, School of Computer Science and Technology, Soochow University, Suzhou, China

[2] King Abdullah University of Science and Technology, Jeddah, Saudi Arabia

[3] Macquarie University, Sydney, Australia

[4] IFLYTEK Research, Suzhou, China

[5] State Key Laboratory of Cognitive Intelligence, IFLYTEK, Hefei, China