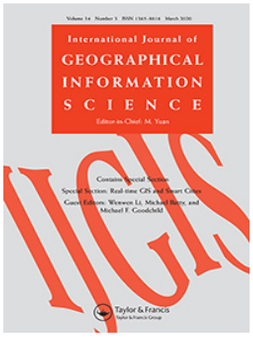

International Journal of Geographical Information Science

ISSN: 1365-8816 (Print) 1362-3087 (Online) Journal homepage: https://www.tandfonline.com/loi/tgis20

A deep learning architecture for semantic address matching

Yue Lin, Mengjun Kang, Yuyang Wu, Qingyun Du & Tao Liu

To cite this article: Yue Lin, Mengjun Kang, Yuyang Wu, Qingyun Du & Tao Liu (2020) A deep learning architecture for semantic address matching, International Journal of Geographical Information Science, 34:3, 559-576, DOI: 10.1080/13658816.2019.1681431

To link to this article: https://doi.org/10.1080/13658816.2019.1681431

INTERNATIONAL JOURNAL OF GEOGRAPHICAL INFORMATION SCIENCE
2020, VOL. 34, NO. 3, 559–576
https://doi.org/10.1080/13658816.2019.1681431

Taylor & Francis
Taylor & Francis Group

RESEARCH ARTICLE

A deep learning architecture for semantic address matching

Yue Lin [a,b], Mengjun Kang [a], Yuyang Wu [c], Qingyun Du [a] and Tao Liu [d]

[a]School of Resource and Environmental Sciences, Wuhan University, Wuhan, China; [b]Department of Geography, The Ohio State University, Columbus, OH, USA; [c]School of Geography and Information Engineering, China University of Geosciences, Wuhan, China; [d]Faculty of Geomatics, Lanzhou Jiaotong University, Lanzhou, China

ABSTRACT

Address matching is a crucial step in geocoding, which plays an important role in urban planning and management. To date, the unprecedented development of location-based services has generated a large amount of unstructured address data. Traditional address matching methods mainly focus on the literal similarity of address records and are therefore not applicable to the unstructured address data. In this study, we introduce an address matching method based on deep learning to identify the semantic similarity between address records. First, we train the word2vec model to transform the address records into their corresponding vector representations. Next, we apply the enhanced sequential inference model (ESIM), a deep text-matching model, to make local and global inferences to determine if two addresses match. To evaluate the accuracy of the proposed method, we fine-tune the model with real-world address data from the Shenzhen Address Database and compare the outputs with those of several popular address matching methods. The results indicate that the proposed method achieves a higher matching accuracy for unstructured address records, with its precision, recall, and F1 score (i.e., the harmonic mean of precision and recall) reaching 0.97 on the test set.

ARTICLE HISTORY
Received 24 May 2019
Accepted 13 October 2019

KEYWORDS
Geocoding; deep neural network; machine learning; semantic matching; word2vec

1. Introduction

Address matching is an essential component of geocoding. The objective of address matching is to match a queried address with one in the address database, so that the queried address can be transformed into a geographical coordinate and therefore can be located on a map. To date, most of the urban data are stored associated with the corresponding address records rather than geographical coordinates (Edwards *et al*. 2014, Ouma *et al*. 2018, Beyer *et al*. 2019); for example, a certain record in a house property database usually contains the resident's name and street address (instead of the latitude and longitude of his/her household). Therefore, to facilitate the geocoding of urban data and to promote the geospatial management of urban systems, an effective address matching method is necessary.

Currently, two major categories of address matching methods exist: (1) string similarity-based methods, which measure the relevance between two compared address records

CONTACT Mengjun Kang mengjunk@whu.edu.cn

and determine whether they match based on a manually set threshold or by using a certain classifier (e.g. a support vector machine or a random forests) (Charif *et al*. 2010, Sun *et al*. 2013, Murrieta-Flores and Martins 2018); and (2) address element-based methods, which parse the address records, compare the respective address elements, and determine whether they match based on the address element hierarchy and their matching rates (Goldberg *et al*. 2007, Melo and Martins 2017). The existing address matching methods mainly focus on the literal relevance between the compared address records; that is, simply making literal word-to-word or character-to-character comparisons between two address records. These methods are useful when the compared address records are both structured and standard; that is, when both address records are complete and correct with respect to a certain address model. However, when the address records are nonstandard (i.e., expressed in an unconventional way, typically with missing address elements or with redundant information) or have complex structures (i.e., without natural separators between address elements, which is common in Chinese addresses), the above-mentioned methods often fail in effective address matching. In addition, the existing methods are unable to identify the matching relations between two synonymous address records when they have few literal overlaps (e.g. Address A and Address B in Table 1). With the increasing popularity of mobile devices and the rapid development of location-based services, a large volume of address-tagged urban data has emerged. As derivatives of multi-source heterogeneous urban big data, most of these address records do not have standard address structures, and there are also inconsistencies in the expressions of place names (Jiang and Yao 2006). In this case, the traditional address matching methods are inadequate.

To deal with the problem stated above, we consider the address semantics and introduce a deep learning architecture in address matching. Deep learning is a novel branch of machine learning that has been broadly applied in geospatial research since 2016. For instance, deep learning has been used to assist in the understanding of urban geography (He *et al*. 2018, Grekousis 2019), in personalized points of interest (POIs) recommendations (Ding and Chen 2018), in the processing of remote sensing images and street-view pictures (Li and Hsu 2018, Srivastava *et al*. 2018), and in efficient geoparsing (Karimzadeh *et al*. 2019). More importantly, significant progress has been made in natural language processing (NLP), one of the most critical tasks addressed by deep learning. The objective of NLP is to enable computers to understand, analyze and process natural (human) languages. Semantic address matching is essentially an NLP task in deep learning. An address is a string formed by consecutive address elements, in which the arrangement of address elements is constrained by their spatial hierarchy (Kang *et al*. 2015, Li *et al*. 2018). The arrangement principles of address elements vary from country to country (or even from city to city), which has led to multiple address models worldwide

Table 1. Examples of synonymous address records with few literal overlaps.

Address A	Address B
福永龙腾阁1巷2502 (Room 2502, Longtengge 1st Lane, Fuyong)	宝安区福永街道白石厦社区德丰路龙腾阁一巷2号 (2 Longtengge 1st Ln, Defeng Rd, Baishixia Community, Fuyong Residential District, Bao'an District)
深圳市深圳大学 (Shenzhen University, Shenzhen City)	南山区南海大道3688号 (3688 Nanhai Ave, Nanshan District)

(e.g. a general address model in China is 'country + province + subprovince + street + room number', while that in the United States is 'room number + street + state + country'). Address models define the syntax of address strings, and the address syntax makes it possible to adopt the deep learning architecture used for NLP problems to semantic address matching.

The deep learning method proposed in this article is implemented with the following two steps. First, the word2vec model is applied to train and obtain the word vectors of address elements and to convert the input address records into their corresponding vector representations. Second, the enhanced sequential inference model (ESIM), one of the state-of-the-art deep text-matching models, is employed to make local and global inferences between the compared address records (in the vector format) and to determine whether they match. By introducing the deep learning architecture into address matching, this article fills a gap by developing an effective and accurate method for address matching that not only considers the literal similarity between address records but also emphasizes their semantic linkage based on the 'understanding' of addresses: the proposed method is able to obtain high predictive accuracy whether the compared address records are alike, or only have few literal overlaps (e.g. Address A and Address B in Table 1). Therefore, our method can outperform the existing methods for address matching even when the queried address has an unstructured or complex syntax.

The remainder of this paper is organized in five sections. Section 2 introduces the materials used in our study, as well as the data processing procedures. The methodology adopted is demonstrated in Section 3, including the principles of word2vec and the ESIM. The results of our experiments are analyzed in Section 4. Section 5 presents our conclusions and the limitations of this study.

2. Materials

Our experiments were based on a large-scale Chinese address corpus derived from the Shenzhen Address Database. The corpus contains 57,253,694 Chinese address records that were updated in the year of 2018, which covers almost all the address data in Shenzhen, China. We applied a set of data cleaning procedures to this corpus that included (1) removing duplicate address records, (2) removing blank characters and special symbols (e.g. '.', '`', and '/'), and (3) modifying the characters that were incorrectly written in address records. Some examples of the processed address records in the corpus are shown in Table 2.

To employ the address data above for semantic address matching, we created a labelled address dataset based on the processed address corpus, the data structure of which is shown in Table 3. This dataset is formed by 84,474 address pairs and the

Table 2. Examples of address records in the corpus and their corresponding English translations.

Address in Chinese	Corresponding address in English
罗湖区红桂路1038号风格名苑B栋17E	Room 17E, Building B, Landmark House, 1038 Honggui Road, Luohu District
石龙新村八区88号703	Room 703, 88 Shilong New Village 8 District
香梅路1072号缇香名苑7栋15F	Room 15F, Building 7, Tixiang Famous Garden, 1072 Xiangmei Road

Table 3. Data structure of the labeled address dataset.

Column	Description
S_a	The address for query
S_b	The address derived from the processed address corpus
label	Denotes whether S_a and S_b are matched: 1 for matched, 0 for unmatched

Table 4. Examples of some data rows in the labeled address dataset.

S_a	S_b	label
深圳风格名苑B-17E (B-17 Landmark House, Shenzhen)	罗湖区红桂路1038号风格名苑B栋17E (Room 17E, Building B, Landmark House, 1038 Honggui Road, Luohu District)	1
深圳风格名苑B-17E (B-17 Landmark House, Shenzhen)	罗湖区红桂路1028号清庆新村59号 (59 Qingqing New Village, 1028 Honggui Road, Luohu District)	0

corresponding labels. Table 4 presents some examples of the data rows in the labelled dataset. The steps to generate this dataset are described below.

Step 1: Randomly selected a subset of the address corpus, which contains 42,237 address records.

Step 2: Manually converted the selected 42,237 address records into their corresponding S_as in order to simulate the nonstandard and ambiguous addresses one might query in a real scenario. The conversion strategies were based on Tian *et al*. (2016), which are listed below.

- *Synonym replacement*: replacing the place name with its street address, or vice versa; replacing the place name with its alias (e.g. converting 'Room 17E, Building B, 1038 Honggui Road' to 'Room 17E, Building B, Landmark House').
- *Element removal*: removing the district, subdistrict or community of an address; removing either the street address or the place name if they both exist (e.g. converting 'Room 17E, Building B, Landmark House, 1038 Honggui Road, Luohu District' to 'Room 17E, Building B, 1038 Honggui Road').
- *Element addition*: adding redundant information (e.g. converting 'Room 703, 88 Shilong New Village 8 District' to 'near the gatekeeper's office at Room 703, 88 Shilong New Village 8 District').

Step 3: Retrieved two S_bs for each S_a, one is matched with S_a (label = 1), and the other is unmatched (label = 0). The matched S_b is the address in the address corpus that had been converted to S_a, and the unmatched S_b was randomly selected from the address corpus.

Table 5 illustrates several statistical characteristics of the labelled address dataset. The length difference is the difference in the numbers of characters between the address pairs. The Levenshtein distance (Levenshtein 1966) is a metric that measures the similarity between two address strings; a shorter Levenshtein distance indicates a higher degree of similarity between address pairs. Similarly, the Jaccard similarity coefficient (Jaccard 1908) is also an address similarity metric; a higher Jaccard similarity coefficient indicates fewer differences between two address records.

As shown in Table 5, S_a and S_b in 97.8% of the unmatched address pairs are in the same administrative districts, and, meanwhile, S_a and S_b in 79.3% of the unmatched address

Table 5. Statistical characteristics of the labeled address dataset.

Attribute	Value
Total number of address pairs	84,474
Number of matching address pairs	42,237
Number of unmatched address pairs	42,237
Number of unmatched address pairs where S_a and S_b are in the same administrative districts	41,313
Number of unmatched address pairs where S_a and S_b are in the same subdistricts	33,482
Average length difference for all the address pairs	4.29
Average length difference for matching address pairs	3.41
Average length difference for unmatched address pairs	5.16
Average Levenshtein distance for all the address pairs	10.91
Average Levenshtein distance for matching address pairs	6.43
Average Levenshtein distance for unmatched address pairs	15.39
Average Jaccard similarity coefficient for all the address pairs	0.48
Average Jaccard similarity coefficient for matching address pairs	0.70
Average Jaccard similarity coefficient for unmatched address pairs	0.25

pairs are in the same subdistricts; these results suggest that S_a and S_b in most of the unmatched address pairs are geographically closer (or more similar). The average length difference between matched and unmatched pairs is comparable, yet the Levenshtein distance and the Jaccard similarity coefficient do indicate that compared to the addresses in the matched pairs, those in the unmatched pairs are relatively dissimilar in structure.

3. Methodology

In this study, we first trained the vectors of address elements derived from the processed address corpus using word2vec and transformed the labelled address dataset from text to vector format. Then, we separated the dataset into three groups: 70% for training, 10% for development, and 20% for test. Finally, we trained the ESIM using the training set to conduct semantic address matching and fine-tuned the model hyperparameters based on the model's accuracy on the development set. The test set was then used to evaluate the accuracy of the best performing model. Figure 1 displays a flowchart of the methodology implemented in our study.

3.1. Obtaining vector representations of address records

As described in Section 2, the address pairs from the labelled address dataset are all in the text format. Because an address is a set of consecutive address elements (Li *et al*. 2018), a general procedure to transform an address string into a computable format for deep learning is to encode each address element with a unique vector and to convert the address into its corresponding vector representation (Figure 2). An address element in this study can be an address entity name (e.g. 'Shenzhen'), an address model feature (e.g. 'district'), or their combinations.

In this study, word2vec was implemented to encode all possible address elements that appear in the address dataset; consequently, each address record in the dataset can be represented by a combination of the word vectors obtained by word2vec. Word2vec is a typical unsupervised neural network language model (NNLM) and works as a simple fully connected neural network (Mikolov *et al*. 2013a, Le and Mikolov 2014). The principle of word2vec is to model the relationships between a given word and the words in its

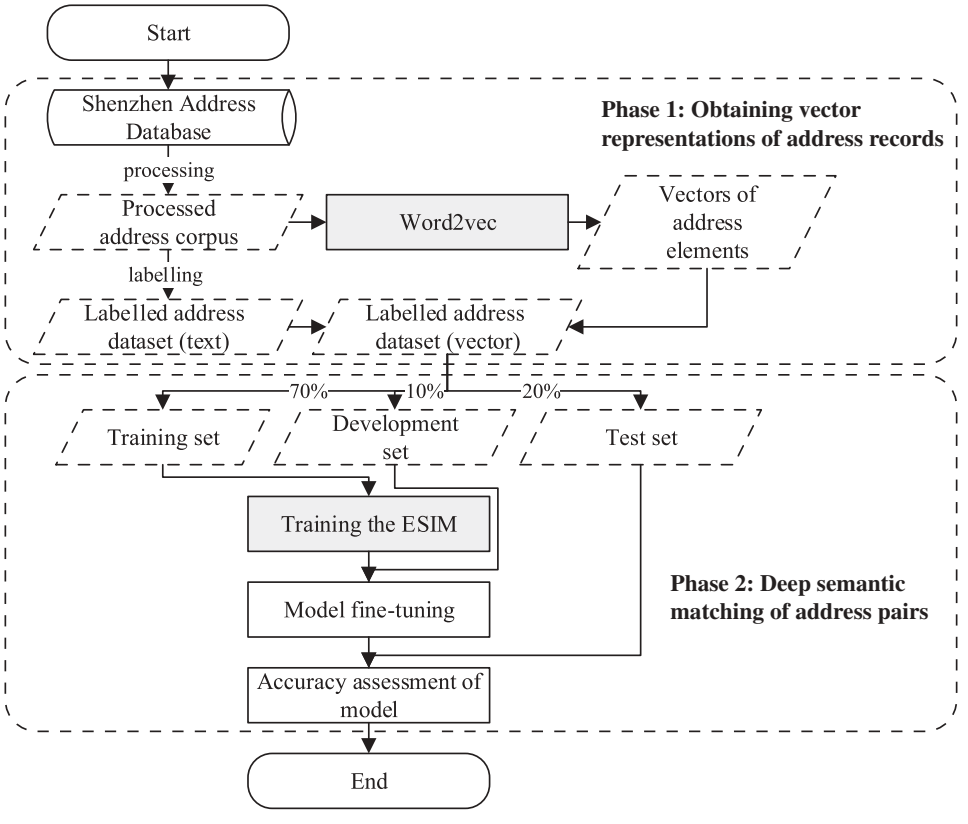

Figure 1. The methodology used in this study.

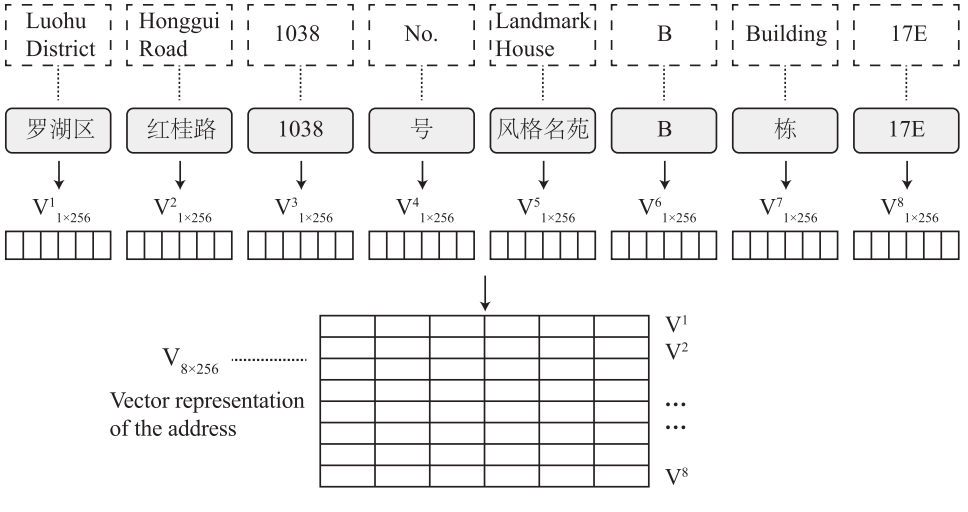

Figure 2. Creating the vector representation of an address.

context and then to make predictions about the subsequent words; therefore, word2vec can take the full advantage of the semantic relations between address elements in an address model when training the word vectors. The objective function of word2vec is to maximize the average log probability $g(w_k)$:

$$g(w_k) = \frac{1}{N}\sum \log p(w_k|Context(w_k))$$

where N denotes the total number of words; w_k denotes the given word, $Context(w_k)$ denotes the words in its context, and $p(w_k|Context(w_k))$ denotes the log probability to predict a word based on its context.

Word2vec has two model architectures: the continuous skip-gram model (Skip-Gram) and the continuous bag-of-words model (CBOW) (Mikolov *et al*. 2013a). The major difference between these two architectures is that Skip-Gram is trained by predicting a given word's context based on the word itself, while CBOW is trained by predicting the given word based on its context. In this study, considering the training efficiency, we applied CBOW as our model architecture (Yu and Dredze 2014, Yao *et al*. 2017). Figure 3 displays the architecture of CBOW.

We employed *Gensim*'s (Řehůřek and Sojka 2010) implementation of word2vec in Python 3.7 to train and obtain the vector representations of address elements. The training corpus we used here was the processed address corpus described in Section 2 because it is a large-scale corpus that covers almost all the address elements in the labelled address dataset. We used negative sampling as our training method, which allows to update only parts of the neural network weights in every training (Mikolov *et al*. 2013b). Moreover, in the training phase, we filtered words with frequencies below 5 to avoid creating too many redundant, low-frequency word vectors; the context window size (i.e., the maximum distance between the given word and the words used to predict it in its context) was set to 10, because a relatively larger context window facilitates the retrieval of the 'local' relations between words (Levy *et al*. 2018). The vector dimension was set to 256 based on the corpus size and the allowable system memory (Levy *et al*. 2018).

Chinese addresses, unlike those in English, do not have natural separators (e.g. blank characters) between elements. If we were to input the raw address records for training, word2vec would recognize each address record as a single word and output a vector of

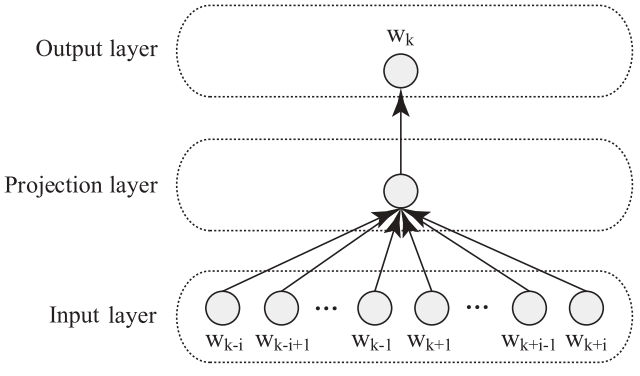

Figure 3. The neural network architecture of word2vec (CBOW) (Mikolov *et al*. 2013a).

Table 6. Examples of tokenized address records and their corresponding English translations.

Tokenized address in Chinese	Corresponding address in English
罗湖区/红桂路/1038/号/风格名苑/B/栋/17E	Room/17E/Building/B/Landmark House/1038/Honggui Road/Luohu District
石龙新村/八/区/88/号/703	Room/703/88/Shilong New Village/8/District
香梅路/1072号/缇香名苑/7/栋/15F	Room/15F/Building/7/Tixiang Famous Garden/1072/Xiangmei Road

the entire address record instead of its elements. Therefore, before training the address vectors, we need to insert separators between address elements in the address corpus; this procedure is also called 'address tokenization'. The *Jieba* library (https://github.com/fxsjy/jieba), a popular tokenization library for Chinese texts based on the directed graphical model (DGM), was utilized for this process, and the tokenization mode was set to 'accurate mode'. To promote the tokenization accuracy, we loaded several gazetteers of Shenzhen as the custom dictionaries. A place name appearing in the dictionaries will be given a higher word frequency during tokenization to reduce its probability of being divided into two or more words. The gazetteers we used here were from the dictionary of Sogou Pinyin (https://pinyin.sogou.com/dict/) and from the Tsinghua University Open Chinese Lexicon (THUOCL) (Han *et al*. 2016). To demonstrate the process, Table 6 displays the tokenization results of the address records from Table 2.

3.2. Deep semantic matching of address pairs

After obtaining the vector representations of address pairs, we employed the ESIM (Chen *et al*. 2016) to conduct semantic address matching. The ESIM is a typical deep learning model for interaction-based text matching (Fan *et al*. 2017). We used the ESIM to make local inference between address pairs and then synthesize the local inference to make a global prediction. This model considers fully the interactions between corresponding address elements and their contexts in two compared address records during model training. An overview of the ESIM implementation is presented in Figure 4.

Assume that S_a (with l_a elements) and S_b (with l_b elements) in the labelled address dataset were both converted to address vectors $a_{l_a \times 256}$ and $b_{l_b \times 256}$ (the vector dimension was defined as 256 in Section 3.1), respectively and were ingested into the ESIM. A detailed illustration of each layer in the ESIM is described below.

Input encoding layer: This layer uses the bidirectional long short-term memory (BiLSTM) model to encode the input address vectors a and and to extract higher-level representations for address records. The BiLSTM model is based on the long short-term memory (LSTM) model (Hochreiter and Schmidhuber 1997) and has an enhanced capability to represent the relationships between a given address element and its long-term context. The BiLSTM model first employs a forward LSTM (initial→end) and then a backward LSTM (end→initial) for a given address vector (a or b); then it concatenates the outputs of both forward and backward LSTMs to generate a new encoding vector $\bar{a}_i(\forall i \in [1, \ \ldots \ , \ l_a])$ or $\bar{b}_j(\forall j \in [1, \ \ldots \ , \ l_b])$. The neural network architecture of the BiLSTM model is shown in Figure 5.

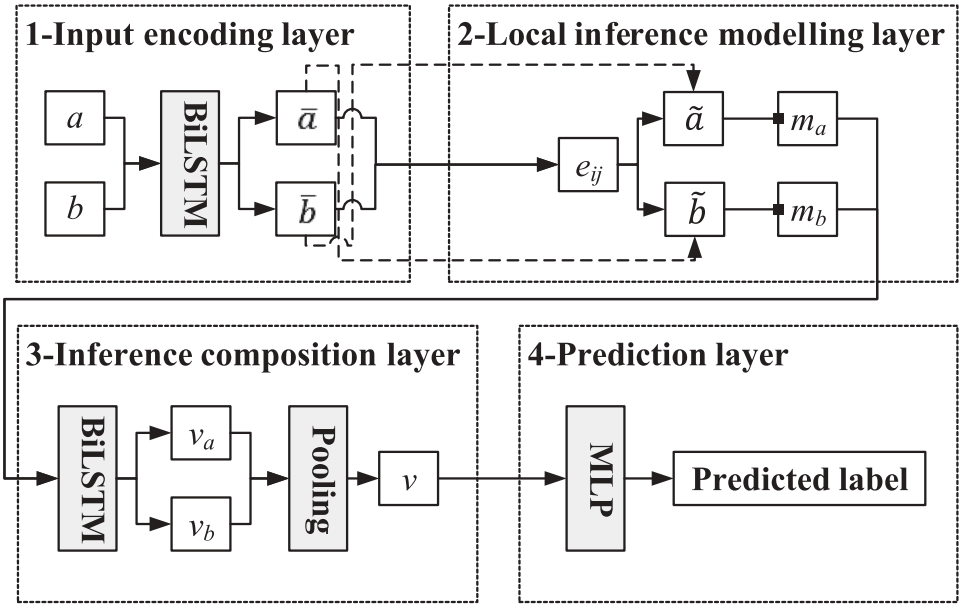

Figure 4. An overview of the ESIM (Chen *et al.* 2016).

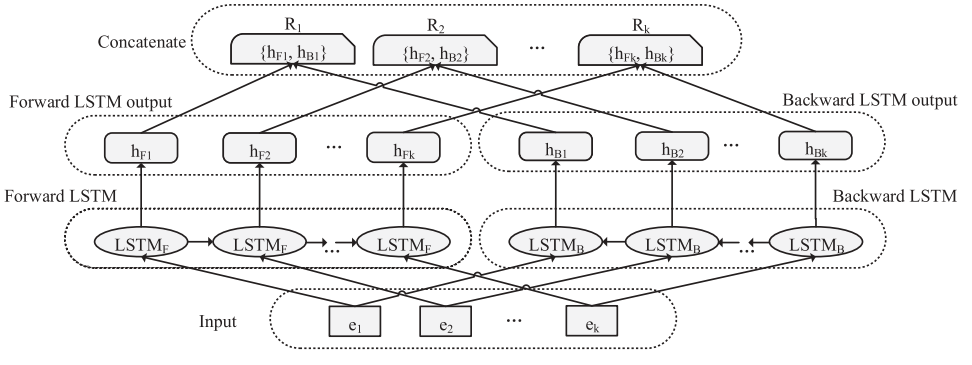

Figure 5. The neural network architecture of the BiLSTM **model (**Chen *et al.* 2016).

Local inference modelling layer: This layer utilizes a modified decomposable attention model (Parikh *et al.* 2016) to make local inference of an address pair using the following three steps: (1) multiply $\bar{a}_i^T$ and $\bar{b}_j$ to obtain an unnormalized attention weight matrix e_{ij}, which represents the local inference between two compared address records; (2) normalize the attention weights using the softmax function (Gibbs 2010), then multiply it by $\bar{b}_j$; the output is represented by $\tilde{a}_i$, and follow the opposite process to obtain $\tilde{b}_j$; and (3) concatenate $\bar{a}$, $\tilde{a}$, their difference and their dot product as m_a to aggregate the local inference of S_a in relation to S_b, and follow the opposite process to obtain m_b.

Inference composition layer: This layer makes global inference between two compared address records on the basis of their local inference with the following two steps: (1)

Table 7. Hyperparameter settings for the ESIM.

Hyperparameter	Description	Experimental setting
Learning rate	The interval for model parameter update in each training epoch; ranging from 0 to 1	Tuned from 0.01, 0.001, to 0.0001
Number of hidden nodes	The number of nodes in the hidden layers	Tuned from 50, 100, to 200
Mini-batch size	The size of training samples used in each epoch; usually ranging from 10 to 100	Tuned from 25, 50, to 75
Dropout probability	The probability for each hidden node to be kept in the dropout; ranging from 0 to 1	0.5
L2 regularization parameter	The value of the L2 regularization term	0.01
Epoch	The number of training epochs	20
Optimizer	The method used to accelerate the convergence rate in stochastic gradient descent (SGD)	Adam optimizer (Kingma and Ba 2015)

extract higher representations of m_a and m_b using the BiLSTM model: the outputs are represented by $v_{a,i}$ and $v_{b,j}$, respectively; and (2) compute both the average and max pooling of $v_{a,i}$ and $v_{b,j}$, and concatenate the results; the pooling step summarizes the local inference and outputs a final vector$v$ with a fixed length.

Prediction layer: This layer utilizes a multilayer perceptron (MLP) to output the predictive results of address pairs (label = 0 or 1). The MLP we used here contains three fully connected layers with rectified linear unit (ReLU), tanh, and softmax activation functions.

This study built the ESIM in TensorFlow 1.13.0 (https://www.tensorflow.org/) using Python 3.7. The model hyperparameters for the ESIM are given in Table 7. To investigate the optimal combination of hyperparameters for the ESIM, we tuned certain hyperparameters within their possible value ranges (Bergstra and Bengio 2012) based on the predictive accuracy on the development set. The hyperparameters we tuned here included the learning rate, the number of hidden nodes, and the mini-batch size, which are generally considered as the most sensitive hyperparameters for deep learning models in most circumstances (Breuel 2015).

The accuracy metrics used in this study include precision, recall, and F1 score (Powers 2011). Precision reflects how many positive labels (i.e., label = 1) are correctly predicted out of the total number of predicted positive labels, and recall reflects how many positive labels are correctly predicted out of the total number of true positive labels. F1 score is the harmonic mean of precision and recall.

4. Results and discussion

4.1. Word vectors of address elements

After word2vec training, we retrieved 159,304 word vectors of address elements with the five most frequent address elements in the address corpus in Table 8. The top three were 'HAO', 'DONG', and 'JIE DAO', which were address model features. The following two were 'SHEN ZHEN SHI' and 'GUANG DONG SHENG', the form of which is the combination of an address entity name and an address model feature. These findings were consistent with Li *et al.* (2018).

Table 8. Examples of the word vectors of address elements.

	Address element (Hanyu Pinyin/ Chinese)	Meaning in English	Word vector
1	'HAO' (号)	Room	[1.6851753, −2.4463012, −0.5596228, −1.008952, −1.4127998, …]$_{1\times256}$
2	'DONG' (栋)	Building	[−1.4107128, −0.08902705, 0.21258928, 0.15118365, −0.4645111, …]$_{1\times256}$
3	'JIE DAO' (街道)	Subdistrict	[−3.772132, 2.4033573, 2.8685732, 0.044730827, −0.6726867, …]$_{1\times256}$
4	'SHEN ZHEN SHI' (深圳市)	Shenzhen city	[3.1039774, 0.7383854, 1.3526704, 1.4857743, −6.2531343, …]$_{1\times256}$
5	'GUAN GDONG SHENG' (广东省)	Guangdong province	[−1.7734634, −1.6311462, 1.2276652, −3.0121398, −1.6291331, …]$_{1\times256}$

4.2. Fine-tuning the ESIM

To attain an optimal set of model hyperparameters for semantic address matching, we fine-tuned the ESIM based on its best predictive results on the development set after 20 epochs of training. Table 9 showed the results of fine-tuning.

As shown in Figure 6, when the learning rate was set to 0.01, as the epoch increased, the training loss fluctuated slightly between the values of 3 and 4, and the F1 score on the development set appeared unstable. When the learning rate was 0.001, the training loss converged in the second epoch, and the F1 score fluctuated around 0.95. When the learning rate was 0.0001, the training loss converged the fastest, and the F1 score remained above 0.96. In addition, as shown in Table 9, when the other hyperparameters

Table 9. Best predictive results of the ESIM and the corresponding time cost under different hyperparameter settings.

	Hyperparameters			Best predictive results			Time cost
	Number of hidden nodes	Learning rate	Mini-batch size	Precision	Recall	F1 score	Training time (h:min:s)
1	100	**0.01**	50	0.95	0.95	0.95	13:43:49
2	100	**0.001**	50	0.96	0.96	0.96	13:41:06
3	**100**	**0.0001**	50	0.98	0.98	0.98	11:24:03
4	**200**	0.0001	50	0.97	0.97	0.97	25:04:07
5	**50**	0.0001	**50**	0.98	0.98	0.98	6:49:46
6	50	0.0001	**25**	0.97	0.97	0.97	11:43:08
7	50	0.0001	**75**	0.98	0.98	0.98	7:22:17

Note: **bold** – the hyperparameter that served as the independent variable of the experiment; underline – the value chosen for the hyperparameter; shading – the hyperparameter combination that was selected as well as the predictive results and time cost under the setting.

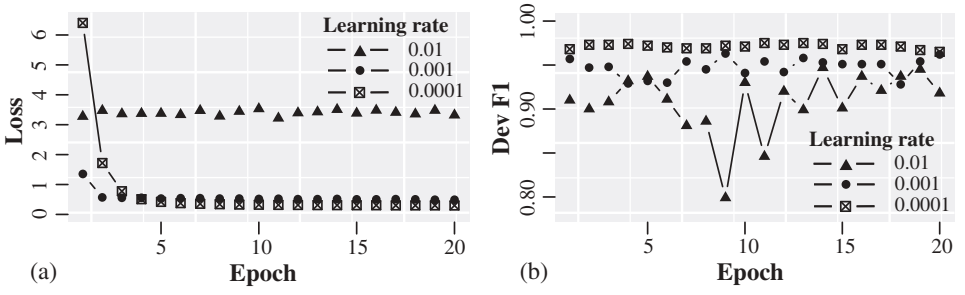

Figure 6. Training losses and F1 scores at different learning rates on the development set.

were held constant, as the learning rate decreased, the time cost for training remained almost unchanged, but all the accuracy metrics of the best predictive results increased. Therefore, we set the learning rate of the ESIM to 0.0001.

As shown in Figure 7, when the number of hidden nodes was set to 50, 100 or 200, the training loss, in all of the three experiments, converged to below 0.5 in the third epoch, and the F1 score on the development set remained above 0.96 most of the time. However, according to Table 9, when the number of hidden nodes was 200, the time cost for training greatly increased, but the best predictive accuracy on the development set decreased, which indicated that the large number of hidden nodes (200) added difficulties to the model training and could lead to overfitting. When the number of hidden nodes was set to 50 or 100, the metrics of the best predictive results were maintained at 0.98, but a smaller value (50) can help to reduce training time. Therefore, we set the number of hidden nodes to 50.

As shown in Figure 8, when the mini-batch size was 25, 50, or 75, the training loss converged to below 0.5 in the seventh epoch. When the mini-batch size was 25 or 50, the F1 score on the development set fluctuated slightly within a small range above 0.96; when the mini-batch size was 75, the amplitude of the F1 score was relatively large, which suggested that a mini-batch size of 75 was too large. Furthermore, as seen in Table 9, when the mini-batch size was 25, the model training took more time, but the best predictive accuracy on the development set decreased, indicating that a mini-batch size of 25 was too small. Therefore, we selected 50 as the mini-batch size in the subsequent experiments.

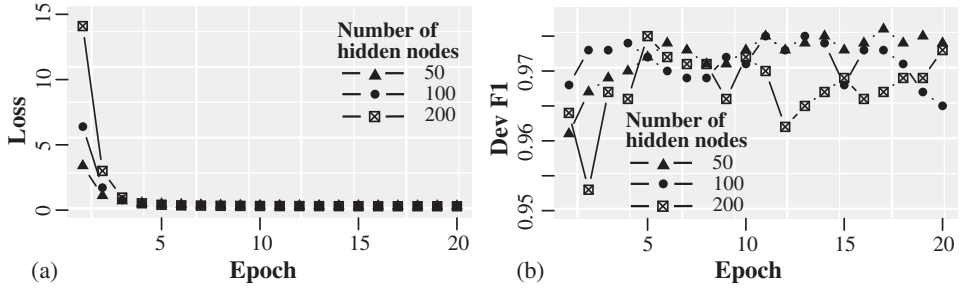

Figure 7. Training losses and F1 scores with different numbers of hidden nodes on the development set.

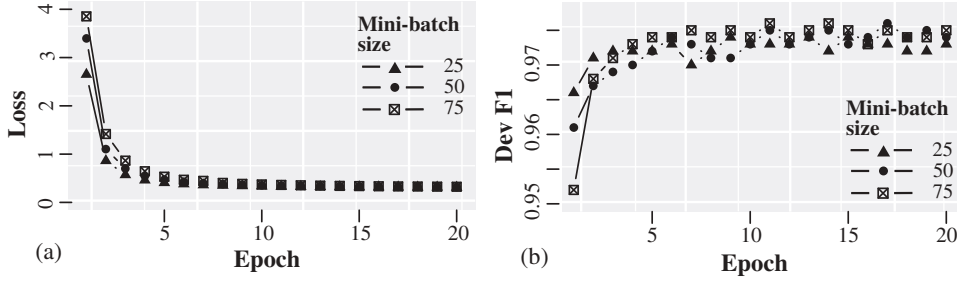

Figure 8. Training losses and F1 scores with different mini-batch sizes on the development set.

4.3. Comparative evaluations of the address matching accuracy

To evaluate the address matching accuracy of the deep learning method proposed in this study, we compared our method with several existing address matching methods based on predictive accuracy on the test set. Our deep learning method used the optimal combination of hyperparameters obtained as explained in the preceding section. The proposed method was first compared with the string similarity-based address matching methods, in which we employed the Levenshtein distance (Levenshtein 1966), the Jaccard similarity coefficient (Jaccard 1908), and the Jaro similarity (Jaro 1989) to measure the string relevance between two compared address records, and at the same time utilized a random forests (RF) classifier (Breiman 2001) and a support vector machine (SVM) classifier (Chang and Lin 2011) to determine whether the address records matched. In addition, we also compared our method with a machine-learning based address matching method proposed by Comber and Arribas-Bel (2019), which utilized a conditional random field (CRF) tokenizer on the address records, applied word2vec to convert the address records into vectors, and made predictions using machine learning classifiers. Furthermore, to demonstrate the effectiveness of the ESIM, we conducted experiments using the word2vec embeddings directly for classification: after training the address vectors, we calculated the cosine similarity between the address pairs and then applied both an RF and an SVM to predict the labels. Table 10 showed the results of the address matching accuracy obtained by these methods.

As shown in Table 10, when using string similarity-based methods for address matching, the Jaccard similarity coefficient can help attain a relatively high predictive accuracy as the classifier remained certain ([3] Jaccard similarity coefficient + RF and [4] Jaccard similarity coefficient + SVM). When the string similarity metrics were fixed, using an RF as the classifier can achieve a high predictive accuracy ([1] Levenshtein distance + RF, [3] Jaccard similarity coefficient + RF and [5] Jaro similarity + RF). Among all of the string similarity-based methods, Jaccard similarity coefficient + RF ([3]) achieved the highest predictive accuracy, with its precision, recall and F1 score all reaching 0.92. However, the deep learning address matching method we introduced in this paper outperformed the Jaccard similarity coefficient + RF ([3]) method: the precision, recall and F1 score of word2vec + ESIM ([11]) on the test set all reached 0.97. Therefore, our proposed method achieved a higher matching accuracy than the traditional text-matching methods did. This finding is consistent with Santos *et al*. (2018).

Table 10. Comparative evaluations of different address matching methods.

		Predictive results		
	Method	Precision	Recall	F1 score
1	Levenshtein distance + RF	0.90	0.81	0.85
2	Levenshtein distance + SVM	0.83	0.88	0.85
3	Jaccard similarity coefficient + RF	0.92	0.92	0.92
4	Jaccard similarity coefficient + SVM	0.91	0.88	0.90
5	Jaro similarity + RF	0.93	0.86	0.89
6	Jaro similarity + SVM	0.96	0.75	0.84
7	CRF + word2vec + RF	0.89	0.85	0.89
8	CRF + word2vec + SVM	0.83	0.76	0.80
9	word2vec + RF	0.92	0.89	0.91
10	word2vec + SVM	0.87	0.81	0.84
11	**word2vec + ESIM**	**0.97**	**0.97**	**0.97**

The machine learning method proposed by Comber and Arribas-Bel (2019) employed a CRF rather than the DGM in address tokenization and utilized a classifier instead of a deep text-matching model to predict the labels. However, compared to the DGM+ word2vec + classifier methods ([9] word2vec + RF and [10] word2vec + SVM), this method showed no improvement in the predictive accuracy. This could be because that the performance of CRF as an undirected graphical model (UGM) was likely to be influenced by the quality of the training corpus (Wallach 2004). Moreover, because the DGM require no global normalization during the training phase, its efficiency is relatively higher than CRF (Steffen Lauritzen 2006). Therefore, it would be rational to use the *Jieba* implementation of the DGM-based tokenizer in this study.

Finally, we compared the methods using word2vec embeddings directly for classification ([9] word2vec + RF and [10] word2vec + SVM) with our proposed method. Although the precision of word2vec + RF ([9]) can reach 0.92, when the ESIM was included in address matching, all the metrics reached 0.97. These results show that the deep text-matching model ESIM improves the predictive accuracy on the test set.

5. Conclusions

In this study, we proposed a deep learning architecture for effective semantic address matching. We first utilized word2vec to convert the input address records from text to vector format and then employed a deep text-matching model, the ESIM, to compute the semantic similarity of the compared address records to determine whether they match. We validated our method using data from the Shenzhen Address Database, a large-scale address corpus. The experimental results showed that the ESIM attained its best results when the number of hidden nodes was 50, the learning rate was 0.0001, and the mini-batch size was 50. In addition, when the queried address was unstructured, the deep learning method introduced in this study outperformed the existing address matching methods in the matching accuracy.

However, this study has some limitations. For example, we did not consider duplicate place names in this paper, because the address database used was limited to a city, where the fuzziness of place names could be ignored. However, if our methodology were applied to a higher-level address database (e.g. a national address database), the model could not resolve two cognominal place names at different locations with the same address elements in the training of word vectors. In addition, when training the vectors of address elements, we did not consider their hierarchal relations; that is, we regarded the address elements at different levels as equally important and assigned the values of word vectors solely based on their word frequency. Therefore, in future work, we plan to implement a language model-based word embedding method, such as Global Vectors (GloVe), to convert address elements into vectors, which might help solve this problem. Moreover, we plan to introduce a strategy to allocate different weights to the address-element vectors based on their hierarchy to further improve the address matching accuracy.

Acknowledgments

We acknowledge Qin Tian for his valuable suggestions on the methodology of this study. We also appreciate the insightful comments from the associate editor, Christophe Claramunt, and all the anonymous reviewers.

Data and code availability statement

The first 498,294 records of the corpus derived from the Shenzhen Address Database, the labelled address dataset for semantic address matching and codes that support the findings of this study are available in Zenodo with the identifiers doi: 10.5281/zenodo.3477633 (part of the corpus for word2vec training), doi: 10.5281/zenodo.3477007 (labelled address dataset for semantic address matching) and doi: 10.5281/zenodo.3476673 (codes). Complete corpus from the Shenzhen Address Database cannot be made publicly available to protect personal information and to follow the national policy on data security.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the National Key Research and Development Program of China [2017YFB0503500].

Notes on contributors

Yue Lin is a first-year PhD student in the Department of Geography at the Ohio State University. She received her bachelor's degree in Geographical Information Science from School of Resource and Environmental Sciences at Wuhan University. Her research interests include GeoAI, spatial data analysis, and urban geography.

Mengjun Kang is an associate professor in School of Resource and Environmental Sciences at Wuhan University. His research interests include geocoding, urban addresses, and digital mapping.

Yuyang Wu is an undergraduate student in School of Geography and Information Engineering at China University of Geosciences.

Qingyun Du is a professor and dean in School of Resource and Environmental Sciences at Wuhan University. His research interests include GIScience and natural language representations of spatial information.

Tao Liu is a professor in Faculty of Geomatics at Lanzhou Jiaotong University.

ORCID

Yue Lin http://orcid.org/0000-0001-8568-7734
Mengjun Kang http://orcid.org/0000-0003-3518-5853
Qingyun Du http://orcid.org/0000-0003-4615-2029

References

Bergstra, J. and Bengio, Y., 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13, 281–305.

Beyer, K., *et al*., 2019. Building capacity for geospatial cancer research in Uganda: a feasibility study. *The Lancet Global Health*, 7, S11. doi:10.1016/S2214-109X(19)30096-8

Breiman, L., 2001. Random forests. *Machine Learning*, 45 (1), 5–32. doi:10.1023/A:1010933404324.

Breuel, T.M., 2015. The effects of hyperparameters on SGD training of neural networks. *arXiv Preprint arXiv:1508.02788*.

Chang, C.C. and Lin, C.J., 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2 (3), 27.

Charif, O., *et al*. 2010. A method and a tool for geocoding and record linkage. *In*: *2010 Second IITA International Conference on Geoscience and Remote Sensing*. Qingdao, China: IEEE, 356–359.

Chen, Q., *et al*., 2016. Enhanced LSTM for natural language inference. *arXiv Preprint arXiv:1609.06038*.

Comber, S. and Arribas-Bel, D., 2019. Machine learning innovations in address matching: A practical comparison of word2vec and CRFs. *Transactions in GIS*, 23 (2), 334–348. doi:10.1111/tgis.12522.

Ding, R. and Chen, Z., 2018. RecNet: A deep neural network for personalized POI recommendation in location-based social networks. *International Journal of Geographical Information Science*, 32 (8), 1631–1648. doi:10.1080/13658816.2018.1447671.

Edwards, S.E., Strauss, B., and Miranda, M.L., 2014. Geocoding large population-level administrative datasets at highly resolved spatial scales. *Transactions in GIS*, 18 (4), 586–603. doi:10.1111/tgis.2014.18.issue-4.

Fan, Y., *et al*., 2017. MatchZoo: A toolkit for deep text matching. *arXiv Preprint arXiv:1707.07270*.

Gibbs, J.W., 2010. *Elementary principles in statistical mechanics: developed with especial reference to the rational foundation of thermodynamics* (Cambridge library collection - mathematics). Cambridge: Cambridge University Press.

Goldberg, W.D., Wilson, J.P., and Knoblock, C.A., 2007. From text to geographic coordinates: the current state of geocoding. *Journal of the Urban and Regional Information Systems Association*, 19 (1), 33–46.

Grekousis, G., 2019. Artificial neural networks and deep learning in urban geography: A systematic review and meta-analysis. *Computers, Environment and Urban Systems*, 74, 244–256. doi:10.1016/j.compenvurbsys.2018.10.008

Han, S., *et al*., 2016. *THUOCL: tsinghua Open Chinese Lexicon* [Online]. Available from: https://github.com/thunlp/THUOCL (Accessed: 24 March 2018)

He, J., *et al*., 2018. Mining transition rules of cellular automata for simulating urban expansion by using the deep learning techniques. *International Journal of Geographical Information Science*, 32 (10), 2076–2097. doi:10.1080/13658816.2018.1480783.

Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. *Neural Computation*, 9 (8), 1735–1780. doi:10.1162/neco.1997.9.8.1735.

Jaccard, P., 1908. Nouvelles Recherches Sur la Distribution Florale. *Bulletin De La Société Vaudoise Des Sciences Naturelles*, 44, 223–270.

Jaro, M.A., 1989. Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. *Journal of the American Statistical Association*, 84 (406), 414–420. doi:10.1080/01621459.1989.10478785.

Jiang, B. and Yao, X., 2006. Location-based services and GIS in perspective. *Computers, Environment and Urban Systems*, 30 (6), 712–725. doi:10.1016/j.compenvurbsys.2006.02.003.

Kang, M., Du, Q., and Wang, M., 2015. A new method of Chinese address extraction based on address tree model. *Acta Geodaetica Et Cartographica Sinic*, 44 (1), 99–107.

Karimzadeh, M., *et al*., 2019. GeoTxt: A scalable geoparsing system for unstructured text geolocation. *Transactions in GIS*, 23 (1), 118–136. doi:10.1111/tgis.v23.1.

Kingma, D. and Ba, J., 2015. Adam: a method for stochastic optimization. *In*: *International Conference for Learning Representations (ICLR)*. San Diego, CA.

Lauritzen, S., 2006. *Graphical Models*. New York: Clarendon Press.

Le, Q. and Mikolov, T., 2014. Distributed representations of sentences and documents. *In*: *International Conference on International Conference on Machine Learning (ICML)*. Beijing, China.

Levenshtein, V., 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10, 707–710.

Levy, O., Goldberg, Y., and Dagan, I., 2018. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3, 211–225. doi:10.1162/tacl_a_00134

Li, L., *et al*., 2018. A hybrid method for Chinese address segmentation. *International Journal of Geographical Information Science*, 32 (1), 30–48. doi:10.1080/13658816.2017.1379084.

Li, W. and Hsu, C.Y., 2018. Automated terrain feature identification from remote sensing imagery: a deep learning approach. *International Journal of Geographical Information Science*, 5 (2–3), 115–141.

Melo, F. and Martins, B., 2017. Automated geocoding of textual documents: a survey of current approaches. *Transactions in GIS*, 21 (1), 3–38. doi:10.1111/tgis.2017.21.issue-1.

Mikolov, T., *et al*., 2013a. Efficient estimation of word representations in vector space. *In*: *International Conference on Learning Representations (ICLR)*. Scottsdale, Arizona.

Mikolov, T., *et al*., 2013b. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 26, 3111–3119.

Murrieta-Flores, P. and Martins, B., 2018. Learning to combine multiple string similarity metrics for effective toponym matching. *International Journal of Digital Earth*, 11 (9), 913–938. doi:10.1080/17538947.2017.1371253.

Ouma, P.O., *et al*., 2018. Access to emergency hospital care provided by the public sector in sub-Saharan Africa in 2015: a geocoded inventory and spatial analysis. *The Lancet Global Health*, 6 (3), e342–e350. doi:10.1016/S2214-109X(17)30488-6.

Parikh, P.A., *et al*., 2016. A decomposable attention model for natural language inference. *In*: *Conference on Empirical Methods in Natural Language* Processing *(EMNLP)*. Austin, Texas.

Powers, D.M.W., 2011. Evaluation: from precision, recall and f-measure to roc., informedness, markedness & correlation. *Journal of Machine Learning Technologies*, 2 (1), 37–63.

Řehůřek, R. and Sojka, P., 2010. Software framework for topic modelling with large corpora. *In*: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: University of Malta, 45–50.

Santos, R., *et al*., 2018. Toponym matching through deep neural networks. *International Journal of Geographical Information Science*, 32 (2), 324–348. doi:10.1080/13658816.2017.1390119.

Srivastava, S., *et al*., 2018. Fine-grained landuse characterization using ground-based pictures: a deep learning solution based on globally available data. *International Journal of Geographical Information Science*. doi:10.1080/13658816.2018.1542698.

Sun, Z., *et al*., 2013. Technology of fuzzy Chinese-geocoding method. *In*: *International Conference on Information Science and Cloud Computing (ISCC)*, Guangzhou, China.

Tian, Q., *et al*., 2016. 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. doi:10.3390/ijgi5050065.

Wallach, H.M., 2004. *Conditional random fields: an introduction*. Philadelphia, PA: University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS -04-21.

Yao, Y., *et al*., 2017. Sensing spatial distribution of urban land use by integrating points-of-interest and Google Word2Vec model. *International Journal of Geographical Information Science*, 31 (4), 825–848. doi:10.1080/13658816.2016.1244608.

Yu, M. and Dredze, M., 2014. Improving lexical embeddings with semantic knowledge. *In*: *Annual Meeting of the Association for Computational Linguistics (ACL)*, Baltimore, Maryland.

Appendix

Table A1. Hardware and software requirements for our deep learning experiments.

Hardware	Setting
Operating system	Windows 10
Central processing unit (CPU)	Intel® Core™ i5-5200U CPU @ 2.20GHz
Random access memory (RAM)	8.00GB
Hard drive capacity	500GB Hard Disk Drive (HDD)
Software	**Setting**
Programming language	Python 3.7
Platform/framework	TensorFlow 1.13.0