

# Detecting Traffic Information From Social Media Texts With Deep Learning Approaches

Yuanyuan Chen<sup>1</sup>, Yisheng Lv<sup>1</sup>, *Senior Member, IEEE*, Xiao Wang<sup>2</sup>, *Member, IEEE*,  
Lingxi Li, *Senior Member, IEEE*, and Fei-Yue Wang<sup>3</sup>, *Fellow, IEEE*

**Abstract**—Mining traffic-relevant information from social media data has become an emerging topic due to the real-time and ubiquitous features of social media. In this paper, we focus on a specific problem in social media mining which is to extract traffic relevant microblogs from Sina Weibo, a Chinese microblogging platform. It is transformed into a machine learning problem of short text classification. First, we apply the continuous bag-of-word model to learn word embedding representations based on a data set of three billion microblogs. Compared to the traditional one-hot vector representation of words, word embedding can capture semantic similarity between words and has been proved effective in natural language processing tasks. Next, we propose using convolutional neural networks (CNNs), long short-term memory (LSTM) models and their combination LSTM-CNN to extract traffic relevant microblogs with the learned word embeddings as inputs. We compare the proposed methods with competitive approaches, including the support vector machine (SVM) model based on a bag of n-gram features, the SVM model based on word vector features, and the multi-layer perceptron model based on word vector features. Experiments show the effectiveness of the proposed deep learning approaches.

**Index Terms**—Deep learning, social transportation, traffic information detection, social media, text mining.

## I. INTRODUCTION

SOCIAL media have evolved dramatically in the past decade, and are now being widely used for posting and sharing user-generated information, ideas, opinions, sentiment, and other forms of expressions [1], [2]. For example, Twitter, a popular microblogging service launched in the USA, has 313 million monthly active users in June 2016 [3]. It generated more than 500 million short messages called “tweet” per day in 2014. In China, Sina Weibo which is akin to Twitter, has

Manuscript received August 10, 2017; revised April 1, 2018 and June 16, 2018; accepted August 30, 2018. This work was supported by the National Natural Science Foundation of China under Grant 61533019, Grant 71232006, and Grant 61233001. The Associate Editor for this paper was L. Li. (*Corresponding author: Yisheng Lv.*)

Y. Chen is with the State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, and also with the University of Chinese Academy of Sciences, Beijing 100190, China (e-mail: yychen5133@ia.ac.cn).

Y. Lv, X. Wang, and F.-Y. Wang are with the State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, and also with the Qingdao Academy of Intelligent Industries, Qingdao 266109, China (e-mail: yisheng.lv@ia.ac.cn; x.wang@ia.ac.cn; feiyue@iee.org).

L. Li is with the Department of Electrical and Computer Engineering, Indiana University–Purdue University, Indianapolis, IN 46202 USA (e-mail: ll7@iupui.edu).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TITS.2018.2871269

over two hundred million registered users and one hundred million active users by the third quarter of 2015. Sina Weibo users post about one hundred million messages called “weibo” or “microblog” per day. Such social media platforms have become powerful and inexpensive information sources due to the huge amount of real-time user-generated contents.

Mining social media data to extract information has gained a lot of attention in a variety of topics. Social media data have been effectively used for natural disaster detection, epidemics monitoring, crisis response and management, and so on. Sakaki *et al.* [4] analyzed the real-time nature of Twitter and proposed an algorithm to monitor tweets and to detect earthquakes with high probability. Bollen *et al.* investigated the correlations between collective mood states derived from Twitter feeds and DJIA (Dow Jones Industrial Average). They found the mood states are predictive of changes in DJIA closing values, and including specific public mood dimension could significantly improve the accuracy of DJIA predictions [5]. Aramaki *et al.* [6] used the support vector machine (SVM) based classifier to extract tweets on actual influenza patients and could detect influenza epidemics with high correlation.

Conventionally, traffic data is collected based on physical sensors like floating cars, closed-circuit television cameras, and loop detectors [7]. Since people and authoritative agencies often post transportation information with the popularity of such platforms, social media have been regarded as the potential source to serve as social sensors to extract traffic information [8], [9]. The term of social transportation was firstly introduced in [10]. Traffic or transportation analytics with social signals using techniques like data mining, parallel intelligence, parallel learning, and natural language processing has recently attracted widespread research interest [11]–[15]. Sasaki *et al.* analyzed the feasibility on detecting transportation information with Twitter, and demonstrated the high potential of using Twitter to detect train status information [16]. Compared to traditional physical traffic sensors, social traffic sensors like Twitter and Sina Weibo have obviously the following advantages: easy access, ubiquitous coverage, free building cost and maintenance cost [10], [17]. They can also provide more insights on traffic information because messages of Twitter and Sina Weibo are usually in the form of texts. Analyzing these messages can help understand a traffic event in terms of where, when, and why it happens.

In this paper, we report our work on extracting traffic relevant short messages from Sina Weibo using convolutional neural networks (CNN), long short-term memory (LSTM)

model, and LSTM-CNN with pre-trained word vectors. Chinese word vectors are learned based on the continuous bag-of-words (CBOW) model using three billion microblogs collected between 2009 and 2011 from Sina Weibo. On top of pre-trained word vectors, we use the CNN and LSTM models to classify microblogs into traffic relevant ones and traffic irrelevant ones. Compared to traditional one-hot word representations and typical feature based text classification methods, the proposed methods can utilize the semantics in microblogs and abstract deep features. Experiments show that the proposed methods have superior performance on extracting traffic relevant microblogs.

The main contributions of this paper are as follows:

- (i) We introduce deep learning methods into extracting traffic information from social media.
- (ii) Different from existing methods that extract traffic relevant microblogs with bag-of-words features, we apply the CBOW model on a dataset of 3 billion microblogs to get the word embedding which captures word semantics.
- (iii) We demonstrate using the deep learning models can help improving detecting traffic related information, compared with the competing models.

The remainder of this paper is organized as follows. In Section II, we review recent advances in social-media based traffic information mining studies. In Section III, we present our approaches mainly including data collection and pre-processing, word vector models, and classification models. In Section IV, we describe experimental validation of the proposed methods. In Section V, we make conclusions and give suggestions for future work.

## II. RELATED WORKS

Social media are quickly becoming ubiquitous and have become one of the main channels to share information. Extracting and analyzing real-time information from social media data is a hot research topic in many fields. In the field of transportation, social media based traffic researches mainly focus on traffic event detection, traffic prediction, and traffic sentiment analysis [18].

Researchers used the features of traffic-related keywords in tweets or microblogs to detect traffic events. D'Andrea *et al.* developed a system based on text mining and machine learning algorithms for real-time traffic event detection from Twitter stream analysis. They firstly collected and manually labeled 1330 tweets consisting of 665 traffic related tweets and 665 non-traffic related tweets. They then built SVM, C4.5, MLP, Naive Bayes and PART models based on bag-of-words features to classify the tweets. Experiments showed the SVM model achieved the best performance. This system was installed and tested for several areas of the Italian road network. It can detect traffic events almost in real time and often before online traffic news web sites and local newspapers [19]. Gu *et al.* proposed a methodology to mine tweet texts to extract incident information on both highways and arterials. They applied an adaptive data acquisition mechanic to collect tweets that were manually labeled with traffic incident label and non-traffic incident label. They adopted a Semi-Naive-Bayes model based on bag-of-words features to classify

the tweets. Then they employed a supervised Latent Dirichlet Allocation model to further classify the traffic incident tweets into five different categories, i.e. accidents, road work, hazards & weather, events and obstacle vehicles. Their approach was applied in the Pittsburgh and Philadelphia Metropolitan Areas in September 2014 [20]. Cui *et al.* [21] developed a prototype system that used Bayesian classifier to firstly filter out traffic related microblogs and then classify these microblogs into traffic flow, traffic accident and traffic control categories. They also designed a question and answering mechanism to ask the user to add necessary information. Gutierrez *et al.* [22] used tweets from regional traffic agencies in UK to detect traffic related events and geo locate the events on a map to notify promptly users. They firstly filtered out traffic related tweets based on the SVM approach and then classified these tweets into 8 classes of traffic event. They also extracted spatial and temporal information by named entity recognition and Part-of-Speech techniques. Tejaswin *et al.* [23] extracted location entities from tweets based on background knowledge from structured data repositories, and then used this data for incident clustering and prediction. Kurkcu *et al.* [24] captured traffic incident information from web-based map providers and Twitter data, and further incorporated incident information into their proposed virtual sensor methodology for automated travel time collection. Zhang *et al.* [25] combined latent Dirichlet allocation and document clustering models to extract incident-level information, and applied pattern analysis to investigate the spatial pattern of incident-topic tweets.

The rich information embedded in online social media data can help improve traffic prediction. He *et al.* [26] addressed the correlation between traffic volume and tweets counts. They proposed a linear regression model incorporating traffic data and Twitter data to predict longer-term traffic flow where the forecasting horizon is beyond 1 hour. Experiments showed that the proposed model outperforms the existing auto-regression based traffic flow prediction model. Ni *et al.* [27] used social media data to develop a short-term traffic flow prediction model under sport game events. They incorporated tweet rate features and semantic features into four prediction models. Experiments demonstrated that including tweet features can improve traffic flow prediction performance. Grosenick [28] extracted non-recurring traffic accidents from twitter data and incorporated this information to predict traffic speed on a single road segment with artificial neural networks. Abidin *et al.* [29] developed a Kalman filter model to predict bus arrival time and analyzed the problem of trust in social media platform. Ni *et al.* addressed the moderate positive correlation between passenger flow and the rates of posted tweets. They extracted event information from tweets, and used both historical transit data and real-time social media data to predict subway passenger flow under event occurrences [30].

There have been some researches using social media data for sentiment and semantic analytics for city traffic. Zeng *et al.* [31] used topic clustering methods to find the attention users giving to various topics concerning Golden Week in China, which can obtain growth tendencies and geographic distributions of travelers. Cao *et al.* proposed a rule-based traffic sentiment analysis system based on web and

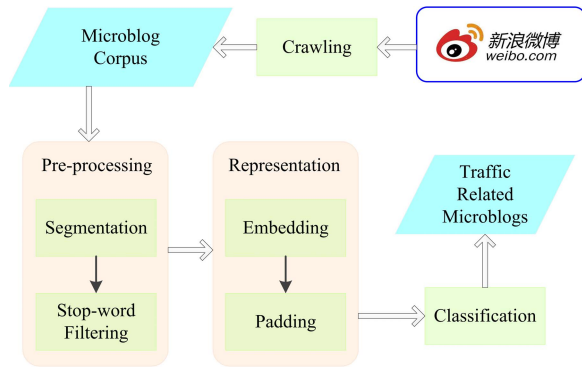


Fig. 1. System flow chart for traffic relevant microblogs detection. The first step is to collect data from social media platform, after which data preprocessing techniques, such as segmentation and stop word removal, are applied. Then the preprocessed texts are transformed into representations that can be handled by machines, and the last step is to train classifiers.

social media data. They demonstrated the efficiency of the proposed methods with two cases in China, i.e. the “yellow light rule” and the “fuel price” in China [32]. Semwal et al. leveraged social media sources such as Facebook, Twitter etc., for event detection, sentiment analysis and suggestion classification [33]. Freddy Lécué et al. proposed a system named STAR-CITY integrating structured and unstructured data, static and stream data, which supports semantic analytics and reasoning for city traffic. They applied semantic web technologies to analyze, diagnose, explore and predict traffic scenarios such as spatial-temporal analysis of traffic status and prediction of road traffic conditions [34].

### III. METHODOLOGY

The methodological framework of this approach mainly includes A. data acquisition by crawling Sina Weibo, B. word segmentation, C. word embedding, D. classification of microblogs to extract traffic information. The whole procedure is illustrated in Fig. 1.

#### A. Data Acquisition by Crawling Sina Weibo

There are two approaches, namely requesting APIs (Application Programming Interfaces) and crawling websites, to access the microblogs in Sina Weibo. Requesting APIs is officially provided for developers, but it is usually not free and has access rate limits. In this paper, we adopted the approach of crawling the Sina Weibo website. This approach is technically free but more comprehensive compared to requesting official APIs.

To fetch raw microblogs, we first set search criteria (e.g., geographic coordinates, time range, keywords). Then we send HTTP requests to the server and obtain the response results which are in the format of HTML source codes. By applying regular expressions matching, we extract microblogs containing the microblog id, the user id, the timestamp, the geographic coordinates and the text from the response results. Then the structured data are stored in a database for future retrieval.

#### B. Word Segmentation

The space is a natural word delimiter for English and many other languages. However, there is no such separator between

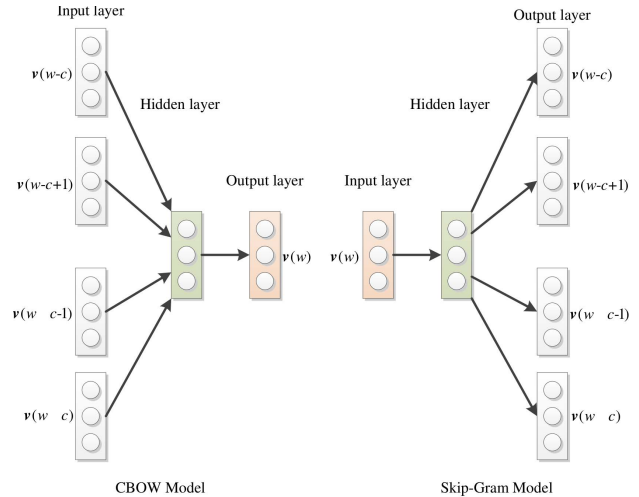


Fig. 2. Model architectures of CBOW and Skip-gram. These two models are mirror images of each other. The learning objective of the CBOW model is to learn word vector to predict the centered word under a context, and the Skip-gram model tries to learn word vector to predict surrounding words based on the centered word.

words in Chinese texts. Therefore word segmentation is the very first task in Chinese language processing. The accuracy of word segmentation is essential to Chinese language mining and understanding. There have been some open source tools on segmenting Chinese words [35], among which we use ICTCLAS (Institute of Computing Technology, Chinese Lexical Analysis System), a well-known and widely used Chinese lexical analyzer, in this paper.

#### C. Word Embedding

Natural language processing systems traditionally represent each word as a one-hot vector which is a vector filled with 0s, except with 1 at the position associated with the word. The one-hot representation is very high-dimensional and sparse. Moreover, such representation cannot capture semantic similarity between words. Representing words in a continuous vector space has been proved effective in natural language processing tasks by grouping similar words. Recently, using neural networks to get the word representations is very attractive and interesting because the learned vectors explicitly encode many linguistic regularities and patterns. The CBOW model and the Skip-gram model are two ways of learning word embedding representations [36], [37]. These two models are similar, except that the learning objective between CBOW and Skip-gram is different. As is illustrated in Fig. 2, the learning objective of the CBOW model is to find word vector representations that are useful for predicting the middle word under a context, while the Skip-gram model tries to learn word vector representations by maximizing the probability of predicting surrounding words based on the middle word.

As the CBOW model and the Skip-gram model are mirror images of each other, we herein just present the derivation of the CBOW model. Due to high computational complexity of the original CBOW model and the Skip-gram model, hierarchical softmax or (and) negative sampling are applied in the training processes. Hierarchical softmax uses a Huffman

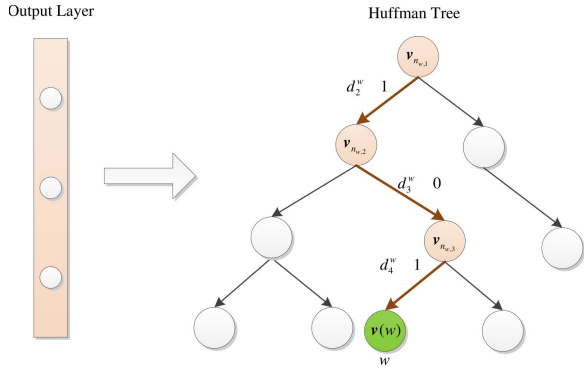


Fig. 3. Output layer of CBOw replaced by a Huffman Tree. The leaf nodes represent the vectors of words in the corpus and one path from the root to node of word  $w$  is highlighted.

tree, a binary tree, to represent all the words in the vocabulary, which are leaf units. In the CBOw model with hierarchical softmax, the output layer is replaced by a Huffman Tree as shown in Fig. 3. And the hidden layer is designed to average the input word vectors, so that the output of hidden layer is

$$\mathbf{h} = \frac{1}{C} \sum_{u \in \text{context}(w)} \mathbf{v}(u) \quad (1)$$

where  $\mathbf{v}(u)$  represents the vector of the word  $u$ ,  $\text{context}(w)$  is the set of contextual words of the word  $w$ , and  $C$  is the cardinality of the set  $\text{context}(w)$ . Given the context, the conditional probability of the word  $w$  is defined as:

$$p(w|\text{context}(w)) = \prod_{j=1}^{L(w)-1} \llbracket \mathbf{h}^T \mathbf{v}'_{n_{w,j}} \rrbracket \quad (2)$$

where  $n_{w,j}$  is the  $j^{\text{th}}$  inner point from the root to word  $w$  in the Huffman tree,  $\mathbf{v}'_n$  is the vector of inner point  $n$ ,  $L(w) - 1$  is the length of the path in Huffman tree for word  $w$ , and  $\llbracket \cdot \rrbracket$  is a function defined as:

$$\llbracket x \rrbracket = \sigma(x)^{d_{j+1}^w} \left[ 1 - \sigma(x)^{(1-d_{j+1}^w)} \right] \quad (3)$$

where  $d_{j+1}^w$  is the  $j^{\text{th}}$  bit of Huffman code for word  $w$ . It is straightforward to train the neural network by maximizing the conditional probability in (2) for a target word  $w$ . Take the logarithm of the conditional probability and define the loss function:

$$L = \log p(w|\text{context}(w)) \quad (4)$$

Then we obtain the derivative of  $L$  with regard to vector of inner point  $n_{w,j}$ :

$$\frac{\partial L}{\partial \mathbf{v}'_{n_{w,j}}} = \frac{\partial L}{\partial \mathbf{h}^T \mathbf{v}'_{n_{w,j}}} \frac{\partial \mathbf{h}^T \mathbf{v}'_{n_{w,j}}}{\partial \mathbf{v}'_{n_{w,j}}} = \mathbf{h}^T \llbracket 1 - \mathbf{h}^T \mathbf{v}'_{n_{w,j}} \rrbracket \quad (5)$$

where  $j = 1, 2, \dots, L(w) - 1$ . And the derivative of  $L$  with regard to vector of contextual words  $u$  is:

$$\frac{\partial L}{\partial \mathbf{v}(u)} = \sum_{j=1}^{L(w)-1} \llbracket 1 - \mathbf{h}^T \mathbf{v}'_{n_{w,j}} \rrbracket \mathbf{v}'_{n_{w,j}} \quad (6)$$

The word vectors are then learnt by maximizing the loss function using the stochastic gradient descent method, which is summarized in Algorithm 1.

---

#### Algorithm 1 CBOw Training Algorithm

---

**Input:** Huffman Tree  $T\_Huffman$ , learning rate  $\eta$ , training epoch  $max\_epoch$ , word embedding dimension  $D$

**Output:** word embedding  $\mathbf{v}$

- 1: //initialize parameters
  - 2: initialize vectors of words in corpus:  $\mathbf{v}$  and vectors of inner points of  $T\_Huffman$ :  $\mathbf{v}'$
  - 3: **for**  $nb\_epoch = 1$  to  $max\_epoch$  **do**
  - 4: //update vectors of inner points
  - 5: **for**  $j = 1$  to  $L(w) - 1$  **do**
  - 6:  $g = \llbracket 1 - \mathbf{h}^T \mathbf{v}'_{n_{w,j}} \rrbracket$
  - 7:  $\mathbf{e} = \mathbf{e} + g \mathbf{v}'_{n_{w,j}}$
  - 8:  $\mathbf{v}'_{n_{w,j}} = \mathbf{v}'_{n_{w,j}} + \eta \mathbf{h}^T g$
  - 9: **end for**
  - 10: //update vectors of contextual words
  - 11: **for**  $u$  in  $\text{Context}(w)$  **do**
  - 12:  $\mathbf{v}(u) = \mathbf{v}(u) + \eta \mathbf{e}$
  - 13: **end for**
  - 14: **end for**
- 

#### D. Classification Models

In this paper, we test three types of deep learning models, i.e., CNN, LSTM, and their combination LSTM-CNN for microblog text classification. Besides, CNN, LSTM, and LSTM-CNN models have been used for traffic prediction [38]–[40].

1) *CNN for Text Classification*: CNN is a feedforward neural network that consists of convolutional layers interspersed with pooling layers as shown in Fig. 4 [41]–[43]. A convolutional layer aims to learn the region features. The convolution operation can be summarized as moving a filter over the sentence matrix (input map) and computing the dot products as shown in Fig. 5. Convolution with one filter outputs a feature vector. To learn more sophisticated features, there are generally distinct filters to convolve the input map, and all feature vectors are concatenated into a new matrix called feature map that would be passed to a pooling layer.

For the next, we would explain the convolution operation in detail. Given a sentence  $s = w_1 w_2 \dots w_N$  consisting of  $N$  words, among which word  $w$  can be represented by  $V$  dimensional word vector, the sentence  $s$  can be transformed into a  $N \times V$  dimensional matrix  $\mathbf{S}$  called the input map. The filter is a  $M \times V$  dimensional matrix  $\mathbf{K}$ . Considering one filter  $p$  only, the  $i^{\text{th}}$  element of the feature vector is:

$$a_i^p = f(\text{conv}([\mathbf{v}(w_i) \dots \mathbf{v}(w_{i+M-1})], \mathbf{K}^p)) + b^p \quad (7)$$

where  $\mathbf{K}^p$  is filter matrix,  $b^p$  is the bias and  $f$  is the activation function. And the operation  $\text{conv}(\cdot)$  is defined as:

$$\text{conv}([\mathbf{v}(w_i) \dots \mathbf{v}(w_{i+M-1})], \mathbf{K}^p) = \sum_{r=1}^M \mathbf{K}_{(r,:)}^p \mathbf{v}^T(w_{i+r-1}) \quad (8)$$

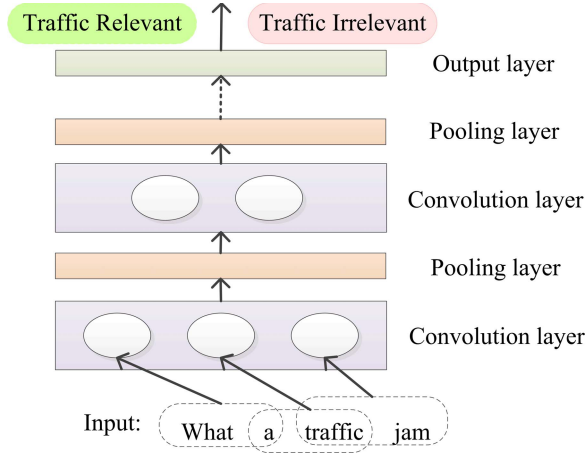


Fig. 4. CNN architecture for text classification. The network consists of convolutional layers with interspersed pooling layers and output layer to give class scores.

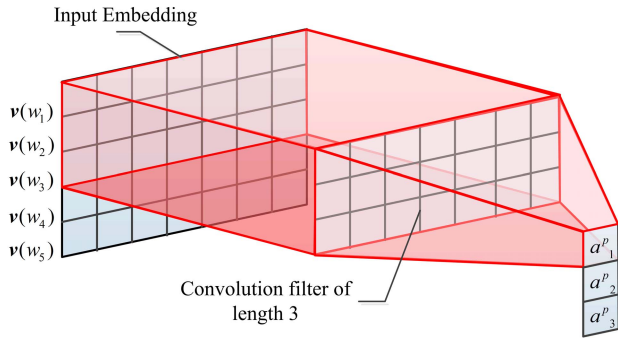


Fig. 5. Convolution for text data. Each convolution unit computes a non-linear function of regional word vectors of input map, where filter parameters matrix and bias are shared by all the units with the same filter.

Substituting 8 into 7, we obtain

$$a_i^p = f \left( \sum_{r=1}^M \mathbf{K}_{(r,:)}^p \mathbf{v}^T(w_{i+r-1}) + b_p \right) \quad (9)$$

Applying (6) repeatedly from  $i$  equals 1 to  $(N - M + 1)$ , then we have the full feature vector  $\mathbf{A}^p = (a_1^p, a_2^p, \dots, a_{N-M+1}^p)^T$  of the convolution layer under the filter  $p$ . If there are  $F$  filters in this case, the dimension of output map is  $(N - M + 1) \times F$ .

The pooling operation computes the average or maximum of each region on the column of the feature map. This operation reduces the dimension of feature map by merging neighboring points, so that computation time is reduced and more abstract information would be passed to the next layers. A pooling layer downsamples the feature map and there will be one output vector for one feature vector. More formally, the output of pooling on feature vector  $\mathbf{A}^p$  is

$$o^p = f(\beta^p \text{down}(\mathbf{A}^p) + b^p) \quad (10)$$

where  $\text{down}(\cdot)$  represents a downsampling function,  $\beta^p$  is the multiplicative bias,  $b^p$  is the additive bias and  $f$  is the activation function. The  $\text{down}(\cdot)$  operation will produce an average or maximum value over  $\mathbf{A}^p$ .

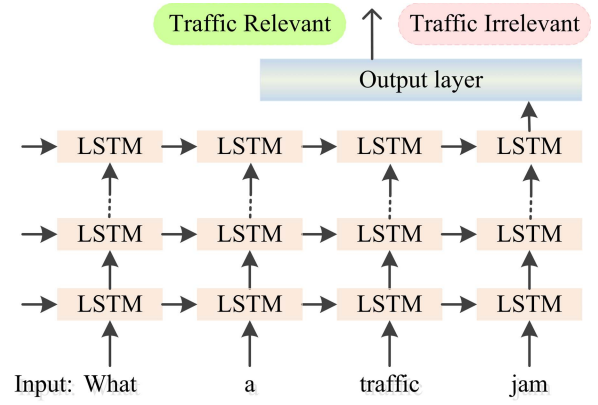


Fig. 6. LSTM architecture for text classification. The network consists of stacked LSTM layers and output layer to give class scores.

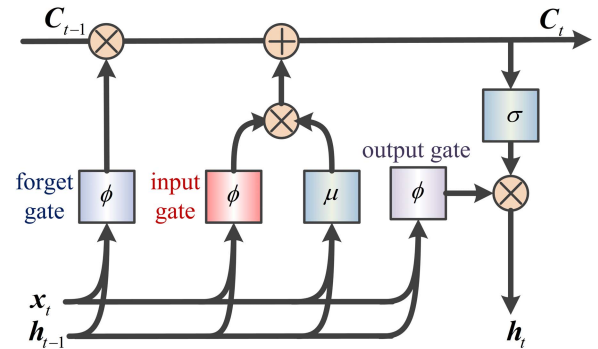


Fig. 7. Architecture of LSTM block. The forget gate, input gate and output gate control the process of adding information to cell state or removing it from cell state.

2) *LSTM for Text Classification*: The LSTM neural network is well-suited to learn from historical experience to predict time series and is widely applied in sequence learning, e.g. speech recognition and traffic prediction [44]–[46]. A LSTM neural network has a feedback loop compared to the feedforward neural network [47], [48]. For the purpose of microblog text categorization, we construct a stacked LSTM neural network. As illustrated in Fig. 6, the LSTM neural network can be unrolled into a series of LSTM blocks, and each block passes state information to a successor. In the stacked LSTM model, the topmost LSTM layer passes the output of the last step to the output layer, while other ones output the full sequence to the next layer.

The LSTM block consists of cell, input gate, forget gate and output gate, as shown in Fig. 7. The gate architecture is for removing information from or adding information to cell state and the cell state is only changed by linear interactions. This design helps to tackle the vanishing gradient problem in standard recurrent neural network as it enables the cell to store and read long range contextual information. To obtain cell state  $C_t$  and hidden state  $h_t$ , we should successively compute the output of the forget gate:

$$f_t = \phi(\mathbf{W}_f[\mathbf{h}_{t-1}, \mathbf{v}(w_t)] + \mathbf{b}_f) \quad (11)$$

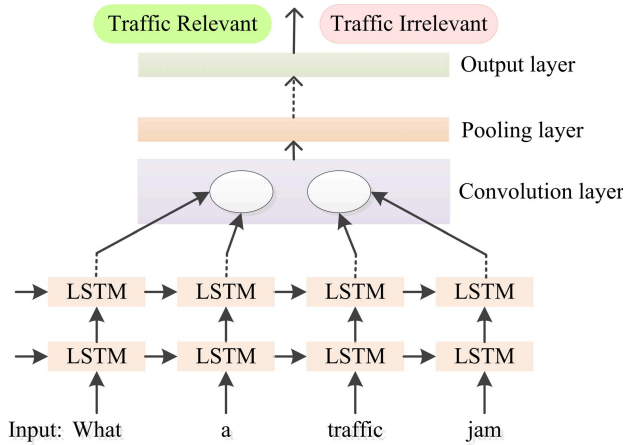


Fig. 8. LSTM-CNN architecture for text classification. The network consists of stacked LSTM layers that return the full sequences, convolutional layers with interspersed pooling layers and fully connected layer as output layer.

the output of the input gate:

$$\mathbf{i}_t = \phi(\mathbf{W}_i[\mathbf{h}_{t-1}, \mathbf{v}(w_t)] + \mathbf{b}_i) \quad (12)$$

and output of the output gate

$$\mathbf{o}_t = \phi(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{v}(w_t)] + \mathbf{b}_o) \quad (13)$$

where  $\mathbf{W}_f$ ,  $\mathbf{W}_i$  and  $\mathbf{W}_o$  are weight matrices of the forget gate, the input gate and the output gate respectively, and  $\mathbf{b}_f$ ,  $\mathbf{b}_i$  and  $\mathbf{b}_o$  are their bias vectors.  $\phi$  is the gate activation function and is usually the sigmoid function. Then compute the cell state:

$$\mathbf{C}_t = \mathbf{f}_t \otimes \mathbf{C}_{t-1} + \mathbf{i}_t \otimes \mu(\mathbf{W}_C[\mathbf{h}_{t-1}, \mathbf{v}(w_t)] + \mathbf{b}_C) \quad (14)$$

and the hidden output:

$$\mathbf{h}_t = \mathbf{o}_t \otimes \sigma(\mathbf{C}_t) \quad (15)$$

where  $\mu$  and  $\sigma$  are activation functions, and they are usually tanh, and  $\otimes$  represents point-wise multiplication. The complete sequence of cell states and hidden outputs can be computed by applying (11), (12), (13), (14) and (15) recursively from  $t = 1$  to  $t = T$ .

3) *LSTM-CNN for Text Classification*: LSTM neural networks have potentials to learn the contextual dependency of a microblog, and CNN can abstract deep features. Thus combining LSTM and CNN would be a promising method to classify microblog sentences. As illustrated in Fig. 8, the input word vectors are fed into the first LSTM layer and the following there may be more LSTM layers. One convolution layer with a pooling layer comes next and also there can be more convolutional layer and pooling layer pairs. Finally the abstract features are feed into a fully connected neural network layer.

#### IV. EXPERIMENTS

##### A. Learning Word Embedding Representations from Unlabeled Microblogs

We use the CBOW model to obtain word embedding representations for further extracting traffic relevant microblogs. The text corpus consists of three billion microblogs covering one million and seventy thousand user accounts. After learning

TABLE I  
KEYWORDS TO SEARCH MICROBLOGS

Chinese Word	English Translation
堵, 拥堵	Congestion
车祸	Traffic accident
剐蹭	Sideswipe
事故	Accident
绕行	Detour
追尾, 相撞	Car Crash
塞车	Jam
路况	Real time traffic

word embedding, each word in the corpus is represented with a word vector and the dimension of this vector is set to 200. The size of the context window is set to 5, namely utilizing 5 words to the left and to the right of a target word. To optimize computational efficiency, we applied hierarchical softmax technique.

##### B. Extracting Traffic Relevant Microblogs

1) *Dataset and Preprocessing*: The dataset for classifying traffic microblogs into traffic relevant and traffic irrelevant ones was collected by searching Sina Weibo with key words that are manually chosen. The key words adopted are given in Table I.

Finally we collect nearly forty thousand candidate microblogs by searching Sina Weibo with key words given in Table I. Then the microblogs are manually labeled with two possible categorizations, i.e. traffic relevant and traffic irrelevant. Among the candidate microblogs, five thousand ones are labeled with the traffic relevant class, and six thousand ones are randomly chosen from the remaining microblogs set which is made up of all candidate ones excluding the traffic relevant ones. The final dataset to train classifiers is made up of five thousand traffic relevant microblogs and six thousand traffic irrelevant ones.

A vocabulary dictionary was built according to the occurrence of words in manually labeled dataset, and each word is represented by its index in the dictionary. Each microblog is transformed into a sequence of word indexes. To feed into deep neural network efficiently, the word index sequence in each microblog for training the classifier is padded into the sequence with the same length. In our experiments, the maximum length of word sequences is 156. And zeroes are padded after each sequence until its length reached 156. As the dimension of word vector is 200, the features of a microblog fill a  $156 \times 200$  matrix, which would be fed into classification models. All the models employed in this paper would utilize this feature matrix as input if there is no additional declaration.

2) *Performance Indexes*: To evaluate the performance of the proposed deep neural network models, we employ three statistical metrics, which are precision, recall and F-measure. These indexes are commonly used in evaluating the classifiers. Table II defines these three indexes. As all these indexes are class specific, for a certain class with label  $l$ , true positive ( $TP$ ) represents the number of instances with label  $l$  correctly assigned with label  $l$ , false positive ( $FP$ ) represents the number of instances with any other labels except  $l$

TABLE II  
PERFORMANCE INDEXES

Index	Definition
Precision	$pre = \frac{TP}{TP+FP}$
Recall	$rec = \frac{TP}{TP+FN}$
F measure	$F_{\beta} = (1 + \beta^2) \frac{pre \cdot rec}{\beta^2 \cdot pre + rec}$

incorrectly assigned with the label  $l$ , and false negative ( $FN$ ) represents the number of instances with label  $l$  incorrectly assigned with any other labels exception  $l$ . Precision measures the exactness of a classifier and recall represents its effectiveness. To get a tradeoff between precision and recall, F-measure is defined as the weighted harmonic mean of precision and recall. When  $\beta$  equals to one, F-measure is also called  $F_1$  score, which is commonly used in evaluating classification models. In what follows, all performance indexes are reported by tenfold cross-validation.

3) *Baseline Methods*: For comparison, we test the SVM model that is reported with good performance for text categorization. One SVM model, named as bow-SVM, takes bag of unigram and bigram as input. And one SVM model named vecseq-SVM, takes the word embedding as input. Both two SVM models were trained using LIBLINEAR library [49]. In bow-SVM based method, we performed feature engineering by combining unigram and bigram features. To experiment with shallow neural network, we also fine-tune multi-layer perceptron method with one hidden layer.

4) *Determination of the CNN Model*: We fix the activation function of convolutional layers to rectifier  $\delta(x) = \max(x, 0)$ , a commonly used activation function in CNN neural networks. And the loss function is defined as binary cross-entropy which is minimized by Adam optimizer [50]. To learn the well-performed classification models, we have conducted a series of experiments to explore the effect of architecture components on model performance. In the experiments of exploring the effect of filter number, the filter number is selected from {10, 20, 50, 100, 150}. According to the results, increasing the number of filters can yield better performance, while it takes longer time to train the model. And the models with 150 filters achieve nearly same performances with the model with 100 filters, which suggests the largest number of filters in grid search can be 150. In the experiments of exploring the effect of filter length, filter length is selected from {2, 3, 4, 5, 6, 10}. According to the results, changing the filter length does not improve performances much for our dataset.

To obtain the best performance model, we run grid search over the filter number and filter length of convolutional layers, and the pooling strategy. The filter number is chosen from {20, 50, 100, 150} and the filter length is set from 2 to 6 with a stride 1. The following pooling operation is chosen from maximum pooling and average pooling. In addition, we have also evaluated the performance of the model by adding a vanilla hidden layer between the top pooling layer and the

TABLE III  
PARAMETER SETTINGS OF THE BEST PERFORMED CNN MODEL

Layer	Type	Filter	Stride	Output Size
0	Input	—	—	(156, 200)
1	Convolution	(4, 200, 100) <sup>1</sup>	1	(153, 100)
2	Max-pooling	(1, 153) <sup>2</sup>	1	(1, 100)
3	Flatten	—	—	(100)
4	Fully Connected	—	—	(50)
5	Fully Connected	—	—	(1)

<sup>1</sup> The filter length is 4, the filter width is 200 (same as the dimension of word embedding), and the filter number is 100.

<sup>2</sup> The horizontal size of pool window is 1 and the vertical size is 153.

TABLE IV  
PARAMETER SETTINGS OF THE BEST PERFORMED LSTM-CNN MODEL

Layer	Type	Filter	Stride	Output Size
0	Input	—	—	(156, 200)
1	LSTM	—	—	(10)
2	Convolution	(5, 200, 10) <sup>1</sup>	1	(152, 10)
3	Max-pooling	(1, 152) <sup>2</sup>	1	(1, 10)
4	Flatten	—	—	(10)
5	Fully Connected	—	—	(1)

<sup>1</sup> The filter length is 5, filter width is 200 (same as the dimension of word embedding), and filter number is 10.

<sup>2</sup> The horizontal size of pool window is 1 and the vertical size is 152.

TABLE V  
PERFORMANCE COMPARISON OF SVM, MLP, CNN,  
LSTM AND LSTM-CNN

Task	Traffic Relevant			Traffic Irrelevant		
	<i>pre</i>	<i>rec</i>	$F_1$	<i>pre</i>	<i>rec</i>	$F_1$
<b>bow-SVM</b>	0.9095	0.8727	0.8908	0.8975	0.9277	0.9123
<b>vecseq-SVM</b>	0.8368	0.7992	0.8176	0.8387	0.8702	0.8542
<b>MLP</b>	0.8752	0.8719	0.8737	0.8997	0.9026	0.9011
<b>CNN</b>	0.8961	0.9006	0.8983	0.9168	0.9130	0.9149
<b>LSTM</b>	0.8873	0.9182	0.9025	0.9298	0.9028	0.9161
<b>LSTM-CNN</b>	0.8928	0.9160	<b>0.9042</b>	0.9285	0.9083	<b>0.9183</b>

top output layer. After running grid search, the model with 100 filters whose length is 4 achieves the best  $F_1$  score, which is summarized in Table III. The best architecture has a maximum pooling layer and a vanilla hidden layer with 50 output units. And the results are given in Table V.

5) *Determination of the LSTM Model*: For the LSTM model, the loss function is also defined as binary cross-entropy. We use RMSprop optimizer [51] to minimize the loss function. To train the model efficiently, we have conducted a series of experiments to explore the effect of the size of LSTM hidden states and the learning rate on model performance. In the experiments of exploring the effect of a learning rate on model performance, the learning rate is selected from 0.01 to 0.05 with a stride 0.01. Experiments show that a lower learning rate gives better performances especially for

recall and  $F_1$  score as the training process going on. In the experiments of investigating the effect of the size of LSTM hidden states on model performance, the size of LSTM hidden state is chosen from  $\{5, 10, 20, 50\}$ . We found that the models with 5 and 10 LSTM hidden state units achieve nearly the same performance after training epoch 30, which are better than that of the models with 20 and 50 LSTM hidden state units.

To get the LSTM model with best performance, a grid search is applied. We choose the size of LSTM hidden states from  $\{5, 10, 20\}$  and the learning rate from 0.01 to 0.03 with a stride 0.01. In addition, we explore whether to stack two LSTM layers would improve the performances. Experiments show that the model with stacked two LSTM layers which have 10 output units and 0.01 learning rate obtains the best  $F_1$  score.

6) *Determination of the LSTM-CNN Model*: In LSTM-CNN model, the first layer is a LSTM layer. And following is a convolutional layer and a maximum pooling layer. The size of LSTM hidden state is chosen from  $\{5, 10, 20, 50\}$ . For convolutional layer, the filter number is chosen from  $\{10, 20, 50, 100\}$  and the filter length is set from 2 to 6 with a stride 1. And the learning rate is chosen from 0.01 to 0.03 with a stride 0.01. After performing grid search, we obtain the best model for  $F_1$  score and the parameter settings are summarized in Table IV.

### C. Results

The detection results of traffic relevant and irrelevant classes with different classification approaches are given in Table V. The  $F_1$  measures of the CNN model, the LSTM model and the LSTM-CNN model are close for both traffic relevant class and traffic irrelevant class, and the LSTM-CNN model achieve the highest  $F_1$  score. All these three models proposed in this paper improve the performance than that of the bow-SVM model, the vecseq-SVM model and the MLP model. It is worth to mention that the bow-SVM is simple yet has a good performance. For traffic relevant class, the precision of the bow-SVM model is the highest compared to that of other methods, but this model has a lower recall which means it would detect fewer traffic relevant microblogs and thus may miss some traffic information such as traffic incidents. It also needs to point that the LSTM model has the highest recall value for traffic relevant class, yet a relatively low precision. A lower precision value means that a model has less exactness and has more false positive instances.

As the performances of bow-SVM, LSTM, CNN and LSTM-CNN approaches are close, we further discuss their benefits and limitations. The proposed deep neural network approach extracts deep features automatically, which is end-to-end learning without manually selecting features compared to the bow-SVM approach. So the deep learning approach needs less expert knowledge on language. Besides, the dimension of feature space in the deep learning approach is much lower than that in the bow-SVM method. In our experiments, the dimension of feature space applied in the bow-SVM method is 276679 that is selected from 1128748 features by dimensionality reduction, while the dimension of feature

space for other methods in this paper is 31200. The limitations of the deep learning approach are also obvious. The end-to-end deep learning, as a black box, lacks model interpretability. Meanwhile, due to high model complexity, the deep learning approach needs much more training time than that of the bow-SVM method.

### V. CONCLUSION

In this paper, we discussed the emerging social transportation topic on extracting traffic information from social media. We proposed deep learning methods to detect traffic related microblogs from Sina Weibo. Different from the traditional n-gram language models taking as input one-hot vector representation of words, we applied word embedding representation of words and used CNN, LSTM, LSTM-CNN to classify short texts of Sina Weibo. We got the word embedding with continuous vector representation of words via the CBOW model using 3 billion microblogs collected between 2009 and 2011 from Sina Weibo. One-hot vector representation of words does not have semantic similarities between words while word embeddings have. The word embeddings were fed into CNN, LSTM, and LSTM-CNN. We investigated the effect of model architecture and hyperparameters on model performance and ran grid search to determine the best architecture of CNN, LSTM, and LSTM-CNN. Experiments showed that the proposed deep learning methods with word embeddings outperform the competing methods.

In future studies, the methodology can be further enhanced and improved in several ways. First, we will collect more microblogs to classify traffic relevant microblogs into detailed categories like traffic accidents, traffic status, etc. Second, the detailed entities, e.g. location and time of occurrence of traffic events, will be extracted from raw microblogs. Furthermore, we will improve short and long term traffic prediction by fusing online social media and traditional physical traffic detector data.

### REFERENCES

- [1] F.-Y. Wang, Y. Yuan, X. Wang, and R. Qin, "Societies 5.0: A new paradigm for computational social systems research," *IEEE Trans. Computat. Social Syst.*, vol. 5, no. 1, pp. 2–8, Mar. 2018.
- [2] F.-Y. Wang *et al.*, "Where does AlphaGo go: From church-turing thesis to AlphaGo thesis and beyond," *IEEE/CAA J. Autom. Sinica*, vol. 3, no. 2, pp. 113–120, Apr. 2016.
- [3] Twitter.com. (Jun. 2016) *Twitter Usage/Company Facts*. [Online]. Available: <https://about.twitter.com/company>
- [4] T. Sakaki, M. Okazaki, and Y. Matsuo, "Tweet analysis for real-time event detection and earthquake reporting system development," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 4, pp. 919–931, Apr. 2013.
- [5] J. Bollen, H. Mao, and X. Zeng, "Twitter mood predicts the stock market," *J. Comput. Sci.*, vol. 2, no. 1, pp. 1–8, Mar. 2011.
- [6] E. Aramaki, S. Maskawa, and M. Morita, "Twitter catches the flu: Detecting influenza epidemics using Twitter," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2011, pp. 1568–1576.
- [7] F.-Y. Wang, "Parallel control and management for intelligent transportation systems: Concepts, architectures, and applications," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 3, pp. 630–638, Sep. 2010.
- [8] X. Wang, K. Zeng, X.-L. Zhao, and F.-Y. Wang, "Using Web data to enhance traffic situation awareness," in *Proc. 17th Int. Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2014, pp. 195–199.
- [9] F. Qu, Z. Wu, F.-Y. Wang, and W. Cho, "A security and privacy review of VANETs," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 6, pp. 2985–2996, Dec. 2015.
- [10] F. Y. Wang, "Real-time social transportation with online social signals," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 3, pp. 909–914, Jun. 2014.



- [11] F. Y. Wang, "Transportation games for social transportation," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 3, pp. 1061–1069, Jun. 2015.
- [12] F.-Y. Wang, X. Wang, L. Li, and L. Li, "Steps toward parallel intelligence," *IEEE/CAA J. Autom. Sinica*, vol. 3, no. 4, pp. 345–348, Oct. 2016.
- [13] W. Chen, F. Guo, and F. Y. Wang, "A survey of traffic data visualization," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 6, pp. 2970–2984, Jun. 2015.
- [14] N.-N. Zheng, S. Tang, H. Cheng, Q. Li, G. Lai, and F. W. Wang, "Toward intelligent driver-assistance and safety warning system," *IEEE Intell. Syst.*, vol. 19, no. 2, pp. 8–11, Mar. 2004.
- [15] F.-Y. Wang, J. Zhang, Q. Wei, X. Zheng, and L. Li, "PDP: Parallel dynamic programming," *IEEE/CAA J. Autom. Sinica*, vol. 4, no. 1, pp. 1–5, Jan. 2017.
- [16] K. Sasaki, S. Nagano, K. Ueno, and K. Cho, "Feasibility study on detection of transportation information exploiting Twitter as a sensor," in *Proc. 6th Int. AAAI Conf. Weblogs Social Media*, 2012, pp. 30–35.
- [17] X. Zheng *et al.*, "Big data for social transportation," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 3, pp. 620–630, Mar. 2015.
- [18] Y. Lv, Y. Chen, X. Zhang, Y. Duan, and N. Li, "Social media based transportation research: The state of the work and the networking," *IEEE/CAA J. Autom. Sinica*, vol. 4, no. 1, pp. 19–26, Jan. 2017.
- [19] E. D'Andrea, P. Ducange, B. Lazzerini, and F. Marcelloni, "Real-time detection of traffic from Twitter stream analysis," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 4, pp. 2269–2283, Aug. 2015.
- [20] Y. Gu, Z. S. Qian, and F. Chen, "From Twitter to detector: Real-time traffic incident detection using social media data," *Transp. Res. C, Emerg. Technol.*, vol. 67, pp. 321–342, Jun. 2016.
- [21] J. Cui, R. Fu, C. Dong, and Z. Zhang, "Extraction of traffic information from social media interactions: Methods and experiments," in *Proc. IEEE 17th Int. Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2014, pp. 1549–1554.
- [22] C. Gutiérrez, P. Figuerias, P. Oliveira, R. Costa, and R. Jardim-Goncalves, "Twitter mining for traffic events detection," in *Proc. Sci. Inf. Conf. (SAI)*, Jul. 2015, pp. 371–378.
- [23] P. Tejaswin, R. Kumar, and S. Gupta, "Tweeting traffic: Analyzing Twitter for generating real-time city traffic insights and predictions," in *Proc. 2nd IKDD Conf. Data Sci.*, 2015, p. 9.
- [24] A. Kurkcu, M. Eng, E. F. Morgul, and K. Ozbay, "Extended implementation method for virtual sensors: Web-based real-time transportation data collection and analysis for incident management," in *Proc. Transp. Res. Board 94th Annu. Meeting*, 2015, pp. 27–37.
- [25] S. Zhang, J. Tang, H. Wang, and Y. Wang, "Enhancing traffic incident detection by using spatial point pattern analysis on social media," *Transp. Res. Rec., J. Transp. Res. Board*, no. 2528, pp. 69–77, 2015.
- [26] J. He, W. Shen, P. Divakaruni, L. Wynter, and R. Lawrence, "Improving traffic prediction with tweet semantics," in *Proc. 23rd Int. Joint Conf. Artif. Intell.*, 2013, pp. 1387–1393.
- [27] M. Ni, Q. He, and J. Gao, "Using social media to predict traffic flow under special event conditions," in *Proc. 93rd Annu. Meeting Transp. Res. Board*, 2014, pp. 3314–3315.
- [28] S. Grosenick, "Real-time traffic prediction improvement through semantic mining of social networks," Ph.D. dissertation, School Sci., Technol., Eng., Math., Univ. Washington, Seattle, WA, USA, 2012.
- [29] A. F. Abidin, M. Kolberg, and A. Hussain, "Improved traffic prediction accuracy in public transport using trusted information in social networks," in *Proc. 7th York Doctoral Symp. Comput. Sci. Electron.*, 2014, p. 19.
- [30] M. Ni, Q. He, and J. Gao, "Forecasting the subway passenger flow under event occurrences with social media," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 6, pp. 1623–1632, Jun. 2017.
- [31] K. Zeng, W. Liu, X. Wang, and S. Chen, "Traffic congestion and social media in China," *IEEE Intell. Syst.*, vol. 28, no. 1, pp. 72–77, Jan. 2013.
- [32] J. Cao *et al.*, "Web-based traffic sentiment analysis: Methods and applications," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 2, pp. 844–853, Apr. 2014.
- [33] D. Semwal, S. Patil, S. Galhotra, A. Arora, and N. Unny, "Star: Real-time spatio-temporal analysis and prediction of traffic insights using social media," in *Proc. 2nd IKDD Conf. Data Sci.*, 2015, Art. no. 7.
- [34] F. Lécué, R. Tucker, V. Bicer, P. Tommasi, S. Tallevi-Diotallevi, and M. Sbodio, "Predicting severity of road traffic congestion using semantic web technologies," in *Proc. Eur. Semantic Web Conf.* Cham, Switzerland: Springer, 2014, pp. 611–627.
- [35] H.-P. Zhang, H.-K. Yu, D.-Y. Xiong, and Q. Liu, "HHMM-based Chinese lexical analyzer ICTCLAS," in *Proc. 2nd SIGHAN Workshop Chin. Lang. Process.*, vol. 17, 2003, pp. 184–187.
- [36] T. Mikolov, K. Chen, G. Corrado, and J. Dean. (2013). "Efficient estimation of word representations in vector space." [Online]. Available: <https://arxiv.org/abs/1301.3781>
- [37] X. Rong. (2014). "word2vec parameter learning explained." [Online]. Available: <https://arxiv.org/abs/1411.2738>
- [38] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data," *Transp. Res. C, Emerg. Technol.*, vol. 54, pp. 187–197, May 2015.
- [39] H. Yu, Z. Wu, S. Wang, Y. Wang, and X. Ma, "Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks," *Sensors*, vol. 17, no. 7, p. 1501, 2017.
- [40] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, "Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction," *Sensors*, vol. 17, no. 4, p. 818, 2017.
- [41] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE Trans. Neural Netw.*, vol. 8, no. 1, pp. 98–113, Jan. 1997.
- [42] J. M. R. Bouvrie, "Notes on convolutional neural networks," Tech. Rep., 2006. [Online]. Available: <http://cogprints.org/5869/>
- [43] Y. Kim. (2014). "Convolutional neural networks for sentence classification." [Online]. Available: <https://arxiv.org/abs/1408.5882>
- [44] Y. Chen, Y. Lv, Z. Li, and F.-Y. Wang, "Long short-term memory model for traffic congestion prediction with online open data," in *Proc. IEEE 19th Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2016, pp. 132–137.
- [45] Y. Duan, Y. Lv, Y.-L. Liu, and F. Wang, "An efficient realization of deep learning for traffic data imputation," *Transp. Res. C, Emerg. Technol.*, vol. 72, pp. 168–181, Nov. 2016.
- [46] Y. Lv, Y. Chen, L. Li, and F. Wang, "Generative adversarial networks for parallel transportation systems," *IEEE Intell. Transp. Syst. Mag.*, vol. 10, no. 3, pp. 4–10, 2018, doi: [10.1109/MITS.2018.2842249](https://doi.org/10.1109/MITS.2018.2842249).
- [47] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.
- [48] A. Graves, N. Jaitly, and A.-R. Mohamed, "Hybrid speech recognition with deep bidirectional LSTM," in *Proc. Autom. Speech Recognit. Understand. Workshop*, Dec. 2013, pp. 273–278.
- [49] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, Jun. 2008.
- [50] D. Kingma and J. Ba. (2014). "Adam: A method for stochastic optimization." [Online]. Available: <https://arxiv.org/abs/1412.6980>
- [51] Y. Dauphin, H. De Vries, and Y. Bengio. (2015). "Equilibrated adaptive learning rates for non-convex optimization." [Online]. Available: <https://arxiv.org/abs/1502.04390>



**Yuanyuan Chen** received the B.E. degree in automation from Tongji University, Shanghai, China, in 2013, and the Ph.D. degree in control theory and control engineering from the University of Chinese Academy of Sciences, Beijing, China, in 2018. He is currently an Assistant Professor with the Institute of Automation, Chinese Academy of Sciences.

His research interests include social transportation systems, and data-driven traffic modeling and prediction.



**Yisheng Lv** (M'11–SM'17) received the B.E. and M.E. degrees in transportation engineering from the Harbin Institute of Technology, Harbin, China, in 2005 and 2007, respectively, and the Ph.D. degree in control theory and control engineering from the Chinese Academy of Sciences in 2010. He is currently an Associate Professor with the Institute of Automation, Chinese Academy of Sciences.

His research interests include traffic data analysis, dynamic traffic modeling, and parallel traffic management and control systems.



**Xiao Wang** (M'16) received the B.E. degree in network engineering from the Dalian University of Technology, Dalian, China, in 2011, and the Ph.D. degree in social computing from the University of Chinese Academy of Sciences, Beijing, China, in 2016. She is currently an Assistant Professor with the Institute of Automation, Chinese Academy of Sciences.

She has published more than a dozen of SCI/EI articles, has translated three technical books (English to Chinese), and has served as a Peer Reviewer with a good reputation for the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, the IEEE/CAA JOURNAL OF AUTOMATION SINICA, and *ACM Transactions of Intelligent Systems and Technology*. Her research interests include social transportation, cyber movement organizations, artificial intelligence, and social network analysis.



**Lingxi Li** (S'04–M'08–SM'13) received the B.E. degree in automation from Tsinghua University, Beijing, China, in 2000, the M.S. degree in control theory and control engineering from the Chinese Academy of Sciences, Beijing, in 2003, and the Ph.D. degree in electrical and computer engineering from the University of Illinois at Urbana–Champaign, Champaign, IL, USA, in 2008. Since 2008, he has been with Indiana University–Purdue University, Indianapolis, IN, USA, where he is currently an Associate Professor of electrical and computer engineering.

His research interests include modeling, analysis, control, and optimization of complex systems, intelligent transportation systems and intelligent vehicles, discrete event systems, active safety systems, and human factors. He was a recipient of the Indiana University Trustees Teaching Award in 2012, the Outstanding Editorial Service Award of the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS in 2012, and the IUPUI Prestigious External Awards Recognition in 2013. He served as the Program Chair for the 2011 and 2013 IEEE International Conference on Vehicular Electronics and Safety. He has been serving as an Associate Editor for the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS since 2009.



**Fei-Yue Wang** (S'87–M'89–SM'94–F'03) received the Ph.D. degree in computer and systems engineering from the Rensselaer Polytechnic Institute, Troy, NY, USA, in 1990. He joined the University of Arizona in 1990 and became a Professor and the Director of the Robotics and Automation Lab and Program in Advanced Research for Complex Systems. In 1999, he founded the Intelligent Control and Systems Engineering Center, Institute of Automation, Chinese Academy of Sciences (CAS), Beijing, China, with the support of the Outstanding

Overseas Chinese Talents Program of the State Planning Council and the 100 Talent Program from CAS, and in 2002, was appointed as the Director of the Key Lab of Complex Systems and Intelligence Science, CAS. From 2006 to 2010, he was the Vice President for Research, Education, and Academic Exchanges at the Institute of Automation, CAS. In 2011, he became the State Specially Appointed Expert and the Director of the State Key Laboratory for Management and Control of Complex Systems.

His current research focuses on methods and applications for parallel systems, social computing, and knowledge automation. He has been elected as a fellow of INCOSE, IFAC, ASME, and AAAS. In 2007, he received the National Prize in Natural Sciences of China and the Outstanding Scientist Award by ACM for his research contributions in intelligent control and social computing. He received the IEEE ITS Outstanding Application and Research Awards in 2009, 2011, and 2015, respectively, and the IEEE SMC Norbert Wiener Award in 2014. Since 1997, he has served as the General or Program Chair of over 20 IEEE, INFORMS, ACM, and ASME conferences. He was the President of the IEEE ITS Society from 2005 to 2007, the Chinese Association for Science and Technology, USA, in 2005, the American Zhu Kezhen Education Foundation from 2007 to 2008, and the Vice President of the ACM China Council from 2010 to 2011. Since 2008, he has been the Vice President and the Secretary General of the Chinese Association of Automation. He was the Founding Editor-in-Chief (EIC) of the *International Journal of Intelligent Control and Systems* from 1995 to 2000 and the *IEEE ITS Magazine* from 2006 to 2007, the EIC of the IEEE INTELLIGENT SYSTEMS from 2009 to 2012 and the IEEE TRANSACTIONS ON ITS from 2009 to 2016. He is currently the EIC of the IEEE TRANSACTIONS ON COMPUTATIONAL SOCIAL SYSTEMS and the Founding EIC of the IEEE/CAA JOURNAL OF AUTOMATICA SINICA and the *Chinese Journal of Command and Control*.