A word-building method based on neural network for text classification

Kai Shuang[a,b], Hao Guo[a], Zhixuan Zhang[a], Jonathan Loo[c] and Sen Su[a]

[a]State Key Laboratory of Networking & Switching Technology, Beijing University of Posts and Telecommunications (BUPT), Beijing, P.R.China; [b]Science and Technology on Communication Networks Laboratory, Beijing, P.R.China; [c]School of Computing and Engineering, University of West London, London, UK

ABSTRACT

Text classification is a foundational task in many natural language processing applications. All traditional text classifiers take words as the basic units and conduct the pre-training process (like word2vec) to directly generate word vectors at the first step. However, none of them have considered the information contained in word structure which is proved to be helpful for text classification. In this paper, we propose a word-building method based on neural network model that can decompose a Chinese word to a sequence of radicals and learn structure information from these radical level features which is a key difference from the existing models. Then, the convolutional neural network is applied to extract structure information of words from radical sequence to generate a word vector, and the long short-term memory is applied to generate the sentence vector for the prediction purpose. The experimental results show that our model outperforms other existing models on Chinese dataset. Our model is also applicable to English as well where an English word can be decomposed down to character level, which demonstrates the excellent generalisation ability of our model. The experimental results have proved that our model also outperforms others on English dataset.

KEYWORDS

Convolutional neural network; long short term memory; structure information; text classification; word-building method

Introduction

Deep learning has made a great progress recently and has played an important role in academia and industry. In particular, standard natural language processing (NLP) approaches for entity and relationship extraction are improved (Wang, Ma, Lowe, Feldman, & Schmitt, 2016) and business-aware concept detection by convolutional neural networks is proposed (Chen, Chen, Chen, & Joshi, 2016). Based on deep neural network, new inspirations are brought to various NLP tasks. Recent progress in word representation contributes to the development of lexical semantics (Liang, Paritosh, Rajendran, & Forbus, 2016).

Text classification is a classic topic for NLP, in which one needs to assign predefined categories to free texts. The range of text classification research goes from designing the best features to choosing the most possible machine learning classifiers. A good text classifier can be applied in many applications such as sentiment analysis (also known as opinion mining) (Liu, 2012; Masdeval & Veloso, 2015; Pang & Lee, 2008; Schulz, Mencía, & Schmidt, 2016), web searching and information filtering (Lai, Xu, Liu, & Zhao, 2015). Therefore, it has drawn a lot of attention from both industry

and academic communities. In this paper, we regard the sentiment classification as the main scene and design a neural network–based model to explore how to improve the accuracy of it.

Existing neural network–based language models for classification have achieved promising results (which can be taken as the state-of-the-art) and mostly have two steps. They first do a pre-training process to learn word embeddings from training corpus (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013; Pennington, Socher, & Manning, 2014) (these word vectors can remain static or no-static during the whole process of classification). After the pre-training process, each word in a sentence is represented by a vector. All vectors are fed into recurrent neural network (RNN), convolutional neural network (CNN) or other neural network models to encode the whole sentence and predict the class attribute (Graves, 2012; Kim, 2014; Sundermeyer, Schlüter, & Ney, 2012).

As most of the existing neural network language models always regard the word as a basic and indivisible unit in languages, they obtain word vectors (formalisation denotations of words) from a pre-training process which is based on co-occurrence probabilities between different words. Thus, the vectors contain the syntactic meaning of words. In this paper, we no longer regard the word as an indivisible unit, but instead show a method to decompose each word in different languages, namely, Chinese and English, in order to extract the morphological meaning of the word, which is also called structure information. In Chinese, each word is made up of several Chinese characters normally one to four, we can decompose each Chinese character into a sequence of radicals according to Shi, Zhai, Yang, Xie, and Liu (2015) and Y. Sun, Lin, Yang, Ji, and Wang (2014). For instance, '高兴' (happy) consists of two characters '高', '兴' and '手舞足蹈' (kick up one's heels) consists of four characters '手', '舞', '足', '蹈'. Each Chinese character can be decomposed into a sequence of radicals which are taken as essential elements for the Chinese character. For example, the character '折' (snap) can be decomposed into '扌' and '斤', the character '妈' (mother) can be decomposed into '女' and '马' and the character '汗' (sweat) can be decomposed into '氵' and '干'. Radicals have two main functions: pronunciation and meaning (Peng, Cambria, & Zou, 2017). For example, the radical '女' in '妈' indicates the meaning and the radical '马' in '妈' indicates the pronunciation. As the aim of this work is text classification, we only focus on the radicals that reflect the meaning function.

From the characteristics of Chinese word-building method, we can get two important regulations. The first is that different Chinese characters with the same radical always carry the same meanings. For example, '海' (sea), '河' (river) and '湖' (lake) all have radical '氵' which are all closely related to water. The second is that the same radical may contain different meanings in different circumstances. For example, the radical '辶' has the meaning of movement when in '过' (pass) and '追' (chase) while it refers to the sense of distance when in '远' (far),'这' (this) and '近' (near).

Based on the characteristics of Chinese, we propose a word-building method based on neural network (WMBNN) where CNN is used to leverage raw information for different languages. Some researchers have found that convolutional networks are useful in extracting information from raw signals (LeCun et al., 1989; LeCun, Bottou, Bengio, & Haffner, 1998). In this method, we first employ our word-building method to decompose Chinese words into a sequence of radicals which are then fed into CNN to obtain the morphological meaning of each word in the form of vector. Then, the morphological vectors and the syntactic vectors are concatenated to make up complete word vectors. Then, these complete word vectors are fed into the LSTM followed by a softmax to predict the possibility of each class attribute for each sentence. Within the scope of our knowledge, we are the first to analyse text classification based on the view of word-building method to extract information of the word.

In order to prove the generalisation ability of WMBNN model, experiments are also conducted on English dataset. We decompose an English word into a sequence of characters according to Kim (2014) and X. Zhang, Zhao, and LeCun (2015). The remaining parts are almost the same as that in the Chinese process which will be discussed in the following parts.

To summarise, our contributions are as follows:

- Based on the characteristics of Chinese word-building method, we decompose each Chinese word into a sequence of radicals which can reflect the inner structure information of a word.
- CNN is applied to extract the structure information from a sequence of radicals to generate the morphological word vectors. Then, they are concatenated with syntactic word vectors (generated from pre-training process) and are fed into LSTM to achieve results which outperform the state-of-the-art.
- The capability of WMBNN is further evaluated with English dataset. The experiment results show that the proposed word-building method is applicable to English where an English word is decomposed down to the character level. The experiment results show that our WMBNN model outperforms the state-of-the-art result again.

Related work

Automatic text classification has drawn a lot of attention from industry and academic communities for many years. The machine learning methods such as Native Nayes, Random Forest, Support Vector Machine (SVM) and Logistic Regression (LR) in Sebastiani (2002) and Aggarwal and Zhai (2012) are used for text classification task in early time. They usually represent the text as a vector with a bag-of-word method and weights are generated by term frequency-inverse document frequency (TF-IDF). With the development of neural network technology, neural network–based models are getting more popular for this task (Kim, 2014; Peng et al., 2017; Tang et al., 2014; Zhu, Sobhani, & Guo, 2015). The power of neural network–based model relies on its ability in learning continuous text representation from data without any feature engineering. For sentence-level sentiment classification, traditional studies always have two steps. They first do a pre-training process to learn word embeddings from data (Bengio, Ducharme, Vincent, & Jauvin, 2003; Mikolov et al., 2013; Pennington et al., 2014; Tang et al., 2014), and then apply compositional semantic approaches to compute the vector of a sentence from the vectors of its constituents. Representative compositional approaches to learn sentence representation include CNN (Kalchbrenner, Grefenstette, & Blunsom, 2014; Kim, 2014), LSTM (Li, Luong, Jurafsky, & Hovy, 2015; Tang, Qin, & Liu, 2015), tree-structured LSTM (Tang et al., 2014; Zhu et al., 2015) and so on. In many NLP tasks, neural network models with the pre-training process can always achieve state-of-art results. As the word vectors from a pre-training process are based on co-occurrence between different words, they only contain syntactic information and ignore the information contained in word structure.

Some researchers have changed the attention from words to characters for some applications of NLP and have proposed some models that use the character-level features for language processing. For the part-of-speech tagging (Dos Santos & Zadrozny, 2014) and named entity recognition (Dos Santos & Guimarães, 2015), improvements have been made by representing a word as a concatenation of its word embedding and an output from a character-level CNN, and using the combined representation as features in a Conditional Random Field. For text classification, character-level convolutional network has been proposed without word embeddings (Zhang et al., 2015). It regards the sentence as a whole and extracts useful information from a sequence of characters with multiple convolution and max-pooling layers. For language model, a character-aware neural model that applies convolutional layers with different widths to extract information from character to represent each word achieves results on par with the existing state-of-the-art with 60% fewer parameters (Kim, Jernite, Sontag, & Rush, 2016).

As for Chinese, some researches have found that radical is important for the computational processing of Chinese language (Sun et al., 2014). The reason lies in that characters with the same radical typically have similar semantic meanings and play similar grammatical roles. Sun et al. (2014) have started to decompose Chinese characters to radicals and proposed a radical enhanced

Chinese character embedding. However, they only selected one radical from each character to enhance the embedding. Peng et al. (2017) have proposed the Radical-Based Hierarchical Embedding model that got radical embedding from a skip-gram-based model. They have decomposed characters into radicals and concatenated them in the order from left to right and treat the radicals as the fundamental units in texts and an average embedding vector has been computed to represent each sentence. Finally, they have used traditional machine learning methods like LR and LinearSVC to perform text classification. Shi et al. (2015) have proposed a model called Short-Text Categorisation (STC) that directly transformed a sentence to a sequence of radicals and applied a convolution layer with one plain layer to make the classification. It is clear that the radicals from different characters mixed together in the STC which may bring in much noise.

WMBNN model

Figure 1 presents the overall logic architecture diagram of WMBNN model for Chinese text. It is clear that the WMBNN model can be divided into two parts from the top of view: (i) word vector generator and (ii) sentence vector generator.

For a better understanding of our WMBNN model, a top-level flow is described with an example as follows. The model now takes a Chinese word '经历了' (w_3) as the current input. w_3 is first fed into the block called word vector generator which aims to generate a word vector to represent each Chinese word. When the word w_3 go into the vector generator block, it respectively goes into the above pipeline and the below pipeline. As we can see, there is a syntactic vector generator on the below pipeline which aims to generate a syntactic vector and the syntactic matrix is generated from the pre-training process. Along the above pipelines, there are two main entity blocks. The first is called word decomposer which decomposes each word referring to the WuBi typewriting method(J. Zhang & Deng, 2012). When the word '经历了' goes into the first block, the output is a radical sequence. Then, the radical sequence goes into the second block called morphological vector generator that aims to employ CNN to extract morphological information from the radical sequence. Finally, the syntactic word vector from below pipeline and the morphological word

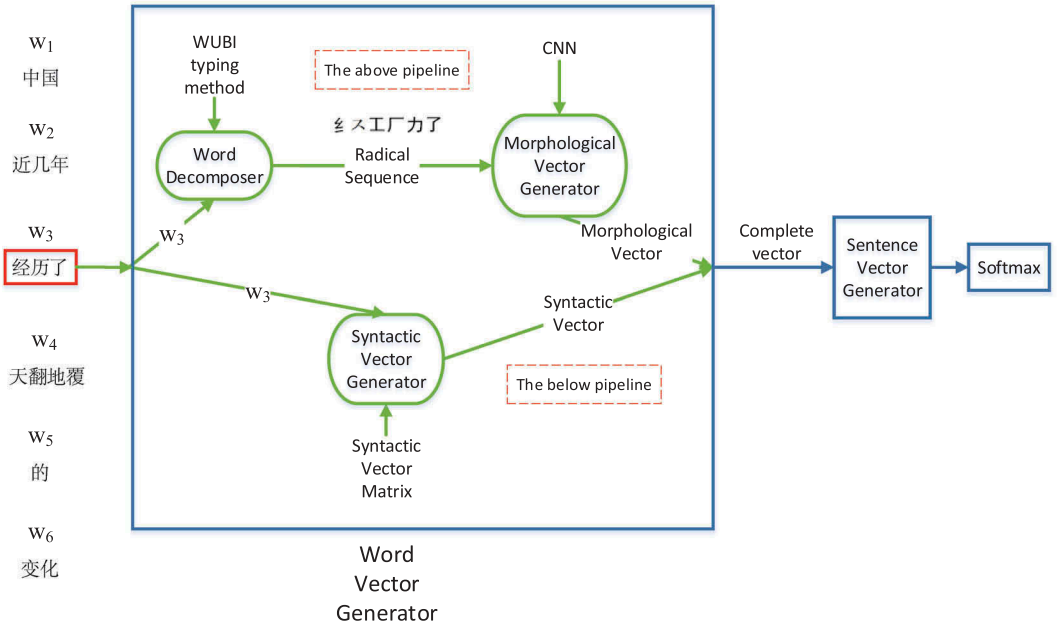

Figure 1. Logic architecture diagram of WMBNN for Chinese.

vector from above pipeline are concatenated together to generate the complete word vector of the w_3. So when a sequence of words goes through the word–vector generator, it will become a sequence of complete word vectors which are fed into sentence vector generator. The sentence generator below employs LSTM to encode the whole vector sequences to one vector where the prediction is made.

Word–vector generator

Preprocessing for Chinese text

As is different from English, there is no blank in a Chinese sentence. So we must do some preprocessing work at first to separate each sentence into several words which is called word segmentation. In our work, we use an open source tool called JieBa (Sun, 2012) to conduct it. After the word segmentation, we can transform the whole sentence into a sequence of Chinese words. For example, when a Chinese sentence '中国近几年经历了天翻地覆的变化' goes through JieBa, it becomes a sequence of six words that are '中国', '近几年', '经历了', '天翻地覆', '的', '变化', respectively. In Chinese, as most of words usually contain one to four characters, when a word exceeds four characters (which is not a common phenomenon), we would just keep the first four characters.

Method of decomposing Chinese words

The word–vector generator aims at generating a distributed representation of each word. In Figure 2, we take the same Chinese sentence used in Figure 1 as an example. To get the syntactic word vector, we can perform a lookup of Chinese word '经历了' embeddings from syntactic matrix X generated during the pre-training process.

Next, the main attention is focused on how to decompose each Chinese character. Radicals are graphical components of characters and the structure of the character can be classified into up–down structure like '夺' or left-right structure like '冰' or semi-enclosed structure like '厉' or full-enclosed structure like '圆'. The radical is the basic unit of the Chinese character and the character structure instructs the way how these radicals are composed together. The concrete way we decompose Chinese character refers to a Chinese typewriting method called WuBi. It decomposes every single Chinese character into most four basic parts according to its structure. Taking the current word '经历了' (w_3) in Figure 2 as an example, it can be decomposed into three Chinese characters '经', '历', '了' in the order from left to right and every Chinese character can be decomposed into a radical sequence. For each Chinese character, we perform a lookup of Chinese radical embeddings (of dimension 15) and stack them to generate the matrix C^m to represent it. Then, we stack these three matrices C^m together according to the sequence they appear in w_3 to obtain the final matrix C^k which contains some essential semantic information of w_3.

Method of generating complete word vectors

To get a morphological vector from matrix C^k, we employ CNN to achieve it. CNN is useful in extracting information from raw signals (LeCun et al., 1989, 1998) and it has also been demonstrated to be effective for various NLP tasks (Collobert et al., 2011). In Figure 2, we can see that there are 150 filters – 50 filters of width four (pink), 50 filters of width five (blue) and 50 filters of width six (green). After the convolution layers, there is a max-over-time pooling operation aimed to obtain a fixed-dimensional representation of the word with dimension 150 (the same as filter numbers). However, different words may have different lengths of the radical sequence. For batch processing, we employ zero padding to make sure that each Chinese character has four radical parts and each word has four Chinese character parts, so the number of columns in C^k is constant for all words. The details of convolution operations are described as follows.

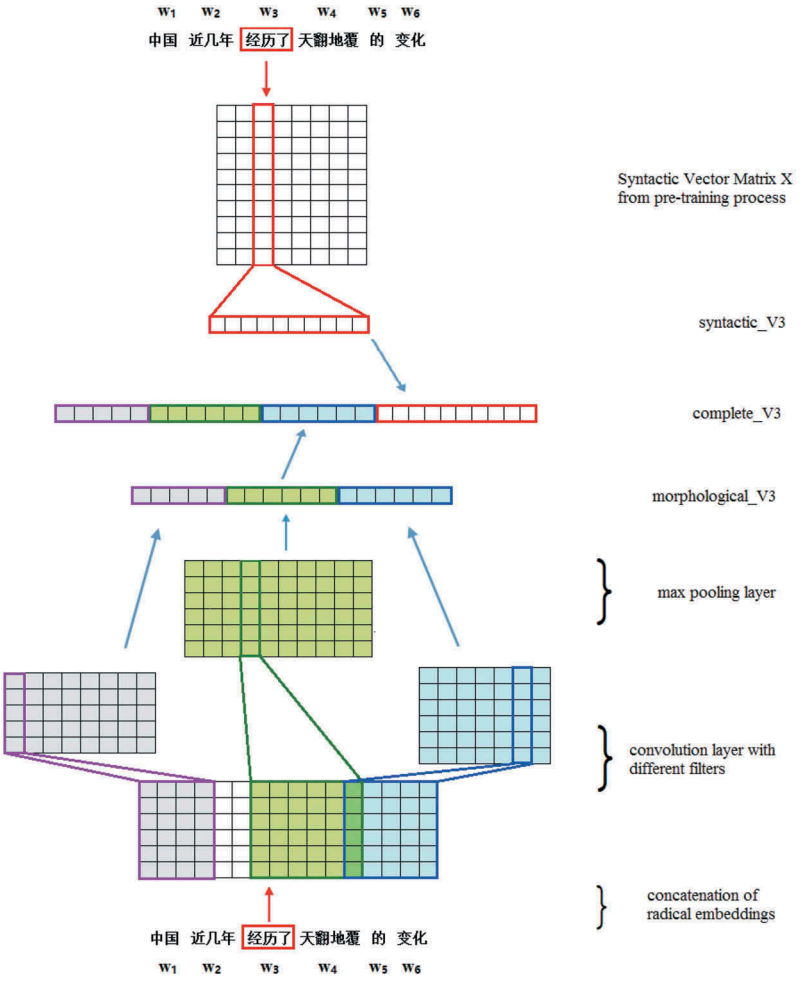

Figure 2. Word–vector generator.

Let d_C be the vocabulary of Chinese radical, d_r be the dimension of Chinese radical embeddings and $Q \in \mathbb{R}^{d_r \times |d_C|}$ be the matrix of Chinese radical embeddings. Assume that word k consists of a sequence of Chinese characters $(c_1, c_2, ..., c_l)$. Then, the word-building method–based representation of k can be transformed into a matrix $C^k \in \mathrm{R}^{4d_r \times l}$, where l is the length of the word k and each character has four-radical space after zero padding operation in our model.

We apply a narrow convolution between C^k and a filter (or kernel) $H \in \mathbb{R}^{d_r \times w}$ of width w, after which we add a bias and apply a non-linearity to obtain a feature map $f^k \in \mathbb{R}^{l-w+1}$. Specifically, the i-th element of f^k is given by:

$$f^k[i] = relu(\langle C^k[*, i : i + w - 1], H\rangle + b) \tag{1}$$

where $C^k[*, i : i + w - 1]$ is the i-to- $(i + w - 1)$-th column of C^k and $\langle A, B\rangle = Tr(AB^T)$ is the Frobenius inner product. Finally, we take the max-over-time

$$y^k = \max_i f^k[i] \tag{2}$$

as the feature corresponding to the filter H (when applied to word k). We can extract the most valuable morphological feature which has the highest score within every filter through this way. A filter can pick out the essential Chinese character n-gram, whose size corresponds to the width of the filter.

The proposed word-building method–based model uses multiple filters of various widths to obtain the feature vector for word k. Suppose that we have a total of h filters $H_1, \ldots, H_h$ and concatenate them together, we can obtain representation of word k in the form of vector that contains morphological information like $y^k = [y_1^k, y_2^k, \ldots, y_h^k]$. Then, it is combined with syntactic information (syntactic word vector from pre-training process) to generate the final complete vector. Thus, when a sequence of words w_1, w_2 and w_3 go through the word vector generator, respectively, we could obtain the complete vector representation of each word at last.

Sentence–vector generator

Some researchers have found that the sentence representation can be naturally considered as the feature to predict the sentiment polarity of the sentence (Graves, 2012; Kim, 2014; Lai et al., 2015; Sundermeyer et al., 2012; Zhou, Sun, Liu, & Lau, 2015). In Figure 3, the output of word vector generator is fed into a LSTM which is a variant of RNN, which is capable of mapping vectors of words with variable length to a fixed-length vector by recursively transforming current word vector v_t with the hidden state vector of the previous step h_{t-1} where we regard the last hidden vector as the sentence representation (J. Li et al., 2015; Tang et al., 2015). Finally, an affine transformation followed by a softmax is applied over the hidden representation of the LSTM to obtain the distribution of its sentiment polarity. Cross-entropy loss between the predicted sentiment polarity distribution over the sentence and the actual sentiment polarity distribution is minimised.

Recurrent neural network

As the sentence–vector generator model, LSTM is one variant of RNN, thus we first introduce some basic notions about RNN.

RNN is a type of neural network model which is more suitable for sequential input data. At each time step t, an RNN takes the input vector $x_t \in \mathbb{R}^n$ and the hidden state vector $h_t \in \mathbb{R}^m$, then produces the next hidden state h_t by applying the following recursive operation:

$$h_t = f(Wx_t + U + b) \tag{3}$$

Here, $W \in \mathbb{R}^{m \times n}, U \in \mathbb{R}^{m \times m}, b \in \mathbb{R}^m$ are parameters of an affine transformation and f is an element-wise non-linearity. However, it is difficult for RNN to learn long-range dependencies due to vanishing/exploding gradients (Bengio, Simard, & Frasconi, 1994). So it can not summarise all the sequence of input and the longer the input sequence is, the more information it will lost.

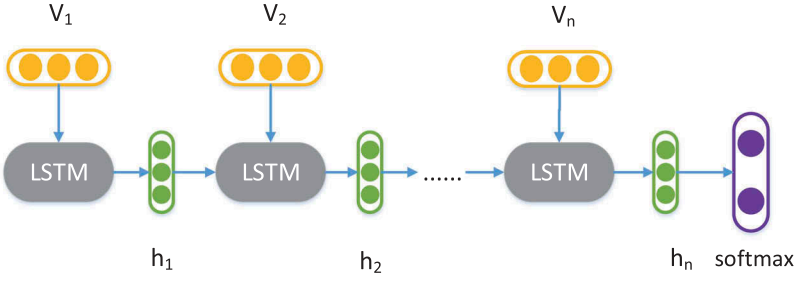

Figure 3. Sentence–vector generator.

Long-short term memory

As RNN suffers from some drawbacks when faced with long sequence input, the internal structure of RNN has to be changed (just has a single memory cell) to solve this problem. One feasible method is adding more memory cells to the model and under the guidance of such a method, the LSTM model was proposed (Hochreiter & Schmidhuber, 1997).

The additive memory cells in LSTM are called forget gate f_t, the input gate i_i and the output gate o_t. These gates collectively determine the transitions of the current memory cell z_t and the current hidden state h_t. Given an input sequence $X = (x_1, x_2, ..., x_n)$, LSTM computes the hidden vector sequence $h = (h_1, h_2, ..., h_n)$ and output vector sequence $Y = (y_1, y_2, ..., y_n)$. Concretely, one step of a LSTM takes x, h, z as input and produces h_t, z_t via the following intermediate calculations:

$$i_t = \sigma(W_i \cdot [h_{t-1}; x_t] + b_i) \tag{4}$$

$$f_t = \sigma(W_f \cdot [h_{t-1}; x_t] + b_f) \tag{5}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}; x_t] + b_o) \tag{6}$$

$$g_t = \tan h(W_r \cdot [h_{t-1}; x_t] + b_r) \tag{7}$$

$$z_t = i_t \odot g_t + f_t \odot z_{t-1} \tag{8}$$

$$h_t = o_t \odot \tan h(z_t) \tag{9}$$

Here, σ is the sigmoid function that has an output in [0, 1], $\tan h$ denotes the hyperbolic tangent function that has an output in [−1, 1] and $\odot$ denotes the component-wise multiplication. The extent to which the information in the old memory cell is controlled by f_t, while i_t controls the extent to which new information is stored in the current memory cell, and o_t is the output based on the memory cell z_t. These memory cells in LSTM help it avoid or alleviate the gradient vanishing problem and suit to learn long-term dependencies. After calculating the hidden vector of each position, we regard the last hidden vector as the sentence representation (J. Li et al., 2015; Tang et al., 2015). We feed it to a linear layer whose dimension of output vector represents class number, and add a softmax layer to output the probability of classifying the sentence as positive or negative. Softmax function is calculated as follows, where E is the number of sentiment categories.

$$\text{softmax}_i = \frac{\exp(x_i)}{\sum_{j=1}^{E} \exp(x_j)} \tag{10}$$

Performance evaluations

First, we implement the neural network models and learning algorithms using the Keras package[1] and scikit-learn package,[2] including the WMBNN, CNN (Kalchbrenner et al., 2014; Kim, 2014), LSTM (J. Li et al., 2015; Tang et al., 2015), Shi et al., 2015), IMBNN (Rossi, Faleiros, Lopes, & Rezende, 2014) and some other traditional learning algorithms to demonstrate that WMBNN model can extract morphological information more effectively from radical sequence which is proved to be helpful to text classification. Second, the rest of the experiments explore the influence of experimental settings in WMBNN model, consisting of the modifications in pre-trained word vectors, the width settings of convolution windows and the number of hidden units. Finally, the same experiments are conducted on English dataset to prove the generalisation ability of WMBNN model.

Performance of WMBNN on Chinese text

To offer a convincing comparison with other models, we conducted a series of experiments with both WMBNN and other models. The source code of our proposed model is available.[3] We will not only prove the effectiveness of WMBNN model but also explore the influence of parameters on WMBNN model in the following experiments.

Chinese dataset

To demonstrate the effectiveness of WMBNN model, we perform the experiments using the following Five datasets in Chinese: Microblog records, Taobao reviews, Fudan Set, Dazhong dataset and Sogou News.

Microblog records. We have crawled them from Sina website[4] and labelled them in two classes that are positive and negative. The total number of microblog records we have crawled is about 1,500,000. Because the records in each class are quite unbalanced, the data prepressing steps are performed by randomly selecting data from the class that contains more records. After the data prepressing steps, we finally get 750,000 records for microblog (380,000 in positive and 370,000 in negative). Here, we call the whole microblog data as Large microblog dataset. Then, we randomly select a subset with about 50,000 records from the Larger microblog dataset as Small microblog dataset for the purpose of analysing the influence of the dataset's size on different models.

Taobao reviews. We have crawled them from Taobao[5] website and labelled them in three classes that are positive, negative and neutral, respectively. The total number of Taobao reviews we have crawled is about 150,000. Similar to the processing methods of Microblog records, we get 100,000 records for Taobao reviews (32,000 in positive, 35,000 in negative and 33,000 in neural).

Fudan set. The Fudan University document classification set is adapted from Lai et al. (2015), which is a Chinese document classification set that consists of 20 classes, including art, education, energy and so on.

Dazhong dataset. We have crawled them from Dazhongdianping[6] website and labelled them in two classes that are positive and negative, respectively. The total number of Taobao reviews we have crawled is about 50,000. Similar to the processing methods of Microblog records, we eventually get 40,000 records for Dazhongdianping dataset (20,000 in positive and 20,000 in negative).

Sogou news. This dataset is constructed according to previews work in Zhang et al. (2015), which contains news articles of five categories – 'sport', 'financeA', 'entertainment', 'automobile' and 'technology'. The number of training samples selected for each class is 90,000 and testing 12,000.

Experimental settings

Word is the most suitable unit in processing Chinese language which we will prove in the following experiments. A Chinese word can be decomposed into one to four Chinese characters and one Chinese character can be decomposed into one to four Chinese radicals which can be regarded as the basic element in the Chinese language. We assign each radical which is generated by the model proposed Shi et al. (2015) with a dimension of 15. In order to identify the boundary of each word, we generate a random vector with a dimension of 15 to represent start-of-word and end-of-word character for that word.

To make the following comparison experiments more convinced and each neural network–based models have the similar model complexity, the hyper-parameters of these neural network–based models are set as follows. The total number of filters in word–vector generator of WMBNN

models are all set as 150 which indicates the dimension of generated morphological word vectors are all 150 (this can make the comparison experiments more reasonable, which will be discussed in next section). We use zero padding to make sure each word embedding matrix has the same length of columns (18 in the experiment) and each sentence has the same words (all have the same length as the longest sentence). The neural network–based models are trained by min-batch backpropagation using optimiser Morop (Tieleman & Hinton, 2012). The learning rate is set 0.001 at the beginning and reduce by half after each two epoch. The training process is stopped when model validation loss does not decrease anymore. The dimensions of LSTM in the following experiments are all set as 150 (This can make the comparison experiments more reasonable which will be described in following part). As for CNN, there are three kinds of filters and the number of them are all 100 with width 2, 3, 4, respectively. The dimension of words is set as 150 in WMBNN models while the rest neural network–based models are set as 300 which are all initialised with Glove or word2vec methods. The radical vectors mentioned above in WMBNN are fine-tuned during the whole training process. The batch size we choose in the experiment is 128 and gradients are averaged over each batch. Parameters of the model are randomly initialised over a uniform distribution with [−0.5, 0.5]. For regularisation, we use dropout (Hinton, Srivastava, Krizhevsky, Sutskever, & Salakhutdinov, 2012) with probability 0.5 on the last softmax layer within all neural network–based models.

In order to find the best hyperparameters for other kinds of comparison model, we apply grid-search approach with 10-fold cross-validation. Finally, the most suitable hyperparameter settings for each model are listed as followsbelow.

For k-earest Neighbor model, k is set as 12 and cosine similarity is used to measure the similarity in k-Nearest Neighbor model. For Random Forest model, number of trees and max-depth are, respectively, set as 500, 6 and Gini impurity is used as the split strategy. For IMBNN model, the error correction rate is set as 0.2 and least mean square is used as the loss function. The smoothing parameter α is set as 1 in Multinomial Naive Bayes. All the above hyperparameters in the comparison models are chosen by grid searching approach.

Comparison models and results

Table 1 shows the accuracy of in different models on five different Chinese datasets. W2V stands for *word2vec* (Mikolov et al., 2013) and Glove stands for *global vector* (Pennington et al., 2014) which are two common word embedding generation strategies aiming at transforming each word into vector space (if the syntactic of two words are similar, the Euclidean distance between them is closer in that space). CNN refers to the model that Yoon Kim has proposed in Kim (2014) and LSTM refers to the model in J. Li et al. (2015) and Tang et al. (2015). WMBNN-without-syntactic-vectors refers to the

Table 1. Accuracy on different Chinese datasets.

Model	Large microblog	Taobao review	Fudan set	Dazhong dataset	Sogou news
WMBNN-W2V	83.23%	72.37%	96.70%	90.23%	92.22%
WMBNN-Glove	83.14%	72.15%	96.92%	90.33%	92.41%
LSTM-W2V	82.43%	70.92%	94.23%	89.37%	91.88%
LSTM-Glove	82.45%	70.99%	94.34%	89.34%	91.98%
CNN-W2V	82.47%	71.13%	94.74%	89.34%	91.45%
CNN-Glove	82.42%	71.20%	94.97%	89.12%	91.71%
WMBNN-without-syntactic-vectors	82.32%	71.10%	95.08%	89.19%	91.33%
STC	79.73%	68.03%	92.10%	77.19%	88.87%
IMBNN	78.87%	67.01%	92.07%	78.93%	89.37%
Bayes	78.11%	66.32%	90.10%	76.53%	84.54%
Wordterm-Randomforest	76.23%	67.33%	92.41%	76.48%	84.54%
Wordvec-LR	75.17%	65.17%	92.58%	73.12%	81.93%
Radicalvec-LR	66.18%	41.46%	91.97%	72.85%	82.83%
Radicalterm-Randomforest	62.23%	40.65%	91.47%	69.09%	77.03%
KNN	63.12%	52.49%	84.95%	66.18%	78.76%

WMBNN model that gives up the syntactic information. WMBNN-W2V refers to the WMBNN model whose syntactic matrix is generated by Word2vec strategy while WMBNN-Glove refers to the one whose syntactic matrix is generated by Glove strategy. STC refers to the model in Shi et al. (2015). IMBNN refers to the model in Rossi et al. (2014). KNN and Bayes refer to the traditional machine learning approaches in Sebastiani (2002) and Aggarwal and Zhai (2012) that only use word frequency information to represent the aim sentence. Radicalvec-LR refers to the Logistic Regression in Sebastiani (2002) and Aggarwal and Zhai (2012), which regards the average of the radical vectors from one sentence as input. Wordvec-LR refers to the Logistic Regression model that regards the average of the word vectors from one sentence as input. Radicalterm-Randomforest refers to the Random Forest model that use bag-of-word method to represent the whole sentence (the elements in the bag are radical from the aim sentence) while the weights of vector are generated by TF-IDF. Wordtem-Randomforest refers to the Random Forest model that uses the bag-of-word method to represent the whole sentence (the elements in the bag are word from the target sentence) while the weights of vector are also generated by TF-IDF. The convolution widths of all filters from word vector generator in WMBNN are set as 8 which will be discussed in next section..

From Table 1, we can see that the accuracies of different models on five datasets have the same tendency. The accuracy of WMBNN-without-syntactic-vectors (82.32% on large microblog, 71.10% on Taobao review, 95.08% on Fudan, 89.19% on Dazhong dataset and 91.33% on Sogou News) is on par with the models with pre-training process, such as CNN (82.47% on Large microblog, 71.13% on Taobao review, 94.74% on Fudan set, 89.12% on Dazhong dataset and 91.45% on Sogou News) and LSTM (82.43% on Large microblog, 70.92% on Taobao review, 94.23% on Fudan set, 89.34% Dazhong dataset and 91.88% on Sogou News). This demonstrates that the WMBNN can effectively extract morphological information from a sequence of radicals which is as important as syntactic information for text classification.

Both WMBNN-W2V (83.23% on Large microblog, 73.37% on Taobao review, 96.70% on Fudan set, 90.23% on Dazhong dataset and 92.22% on Sogou News) and WMBNN-Glove (83.14% on Large microblog, 72.15% on Taobao review, 96.92% on Fudan set, 90.33% on Dazhong dataset and 92.41% on Sogou News) outperform the other models which demonstrates that combining morphological information and syntactic information can further improve the accuracy of classification compared to the traditional CNN and LSTM models using only syntactic information. The STC directly applies convolution layer to extract information from a raw radical sequence, leading the radicals from different characters mixed together, which will bring in much noise for classifcation. The Bayes and IMBNN just consider the co-occurrence between independent words and sentences, but when encountering the sentence containing some negation words or intensity words which can directly change the polarity of the sentence (pretty common in our two datasets) they behave badly. Both Radicalvec-LR and Radicalterm-Randomforest behave terribly compared to other models, because only the composition of all radicals from the same word can hold complete meanings (word is the basic and indivisible in a language) while mixing the radicals in a sentence together would make no sense. As for Wordvec-LR and Wordterm-Randomforest, they all mix the words from one sentence together, so they will lose word sequence information that can influence the accuracy of classification.

Table 2 shows the classification accuracies of WMBNN with different convolution filter width settings on Large microblog dataset and Taobao review dataset. The total number of filters in each model is set as 150 which represents the dimension of morphological word vectors. 3–4–5 means that there are three kinds of filters with the width of 3, 4, 5 respectively and each kind of filter has 50 filters. 8 means that there is only one kind filter with the width of 8. Each model employs Glove strategy to generate syntactic word vectors that are combined with morphological word vectors to generate the complete word vectors.

From Table 2, we can see that different settings of convolution filter width exactly have different effects on the accuracy of WMBNN model. Experiment with 8-width filter achieves the highest accuracy among all settings on all datasets. The width 8 is exactly equal to the summation of the radical embedding number of two Chinese characters (each Chinese character can be decomposed

Table 2. Accuracy of different width settings.

Width settings	Large microblog	Taobao review	Fudan set	Dazhong dataset	Sogou news
8	83.23%	72.37%	96.70%	90.23%	92.22%
7–8-9	83.11%	72.14%	96.34%	90.10%	92.13%
6–7-8	82.95%	71.91%	96.24%	90.10%	92.09%
7	82.92%	72.12%	95.24%	89.98%	91.07%
4–5–6	82.91%	72.02%	95.25%	89.64%	91.46%
3–4-5	82.83%	71.99%	95.17%	89.34%	92.07%
9	82.78%	72.15%	95.23%	89.39%	92.05%
4	82.88%	72.34%	95.31%	89.72%	91.78%
5-6-7	82.72%	71.81%	95.11%	89.19%	91.24%
10	82.72%	72.03%	95.03%	88.59%	91.98%
6	82.70%	72.09%	94.78%	88.77%	90.36%
5	82.65%	71.88%	94.56%	88.42%	90.17%

into four radical embeddings with zero padding). This result may be relevant to the characteristics of the Chinese language. In most cases, two adjacent characters can always convey integrate meanings which are related to this word. When a sequence of adjacent characters in a word are conducted convolutional operations and go through a max-pooling layer, the essential morphological feature of the word would be extracted.

Figure 4 shows the classification accuracy of WMBNN model with different dimensions of the hidden state vector in LSTM. In this experiment, we set the width of all filters as 8 which has been proven to be the optimum choice among many candidates. The axis X represents the dimension of the hidden state vector in LSTM. The convex curve in Figure 4 presents the variation of the classification accuracy with the growth of the dimension of the hidden state vector. The accuracy goes up monotonously at first, because the hidden state vector encodes the whole sentence and longer sentence vector can contain more information of the sentence which can help improve the prediction accuracy. When the dimension of the hidden state vector continues going up and exceeds 200, the accuracy begins to decline. Because when the dimension of the hidden state vector exceeds a critical value (200 in this experiment, which is closely relevant to the data size), the benefit of prediction accuracy from the longer dimension of the hidden state vector is outweighed by the loss produced from the more complexity of this model. It is clear that there is a trade-off between the dimension of the hidden state vector

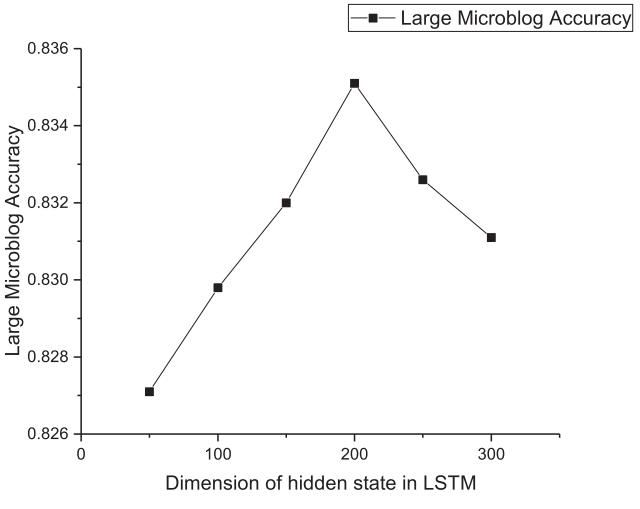

Figure 4. Accuracy of different dimensions of hidden state.

Table 3. Accuracy on small microblog dataset.

Model	Small microblog	Decline rate
WMBNN-W2V	83.14%	0.09%
WMBNN-Glove	83.05%	0.09%
LSTM-W2V	82.21%	0.22%
LSTM-Glove	82.16%	0.29%
CNN-W2V	82.23%	0.24%
CNN-Glove	82.13%	0.29%
WMBNN-without-syntactic-vectors	82.21%	0.11%
STC	79.40%	0.33%
IMBNN	78.15%	0.72%
Bayes	77.26%	0.85%
Wordterm-Randomforest	75.91%	0.32%
Wordvec-LR	74.84%	0.34%
Radicalvec-LR	65.44%	0.74%
Radicalterm-Randomforest	61.35%	0.88%
KNN	62.46%	0.66%

(dimension of sentence vector) and model complexity, and the choice 200 exactly stands at the balance point in experiments.

Table 3 shows the classification accuracy of different models on small microblog dataset and the decline rates compared those in large microblog dataset are also listed above. The terms in Table 3 have the same meaning as those in Table 1.

From Table 3, we can see that the results on small microblog dataset have the same tendency with those in Table 1. However, there are still some important details that we should pay attention to. Firstly, the accuracy of a model declines from large microblog dataset to small microblog dataset in every model. This is because larger dataset contains more information which is helpful for training a complex model. Secondly, when size of the dataset changes, WMBNN (WMBNN-without-syntactic-vectors, WMBNN-W2V, WMBNN-Glove) all have less decline rates of accuracy than the other models. The difference shows that WMBNN are less sensitive to the data size than the other ones.This is because WMBNN model can decompose each word to a number of radicals, and the morphological information it extracts is based on statistics of radicals (one radical can be shared by many different words, so radicals vectors are less sensitive to the data size), which makes the WMBNN model preform more stability facing the changes of data size. As for IMBNN and Bayes model, both of them are based on the co-occurrence between words and different classes of sentences that makes them decline much more when the data size shrinks (the larger dataset is, the more closer the count of co-occurrence is to the true probability distribution).

Performance of WMBNN on English text

Figure 5 presents the overall logic architecture diagram of WMBNN model for English text. We present that only the word decomposer block is different from that in Figure 1 throughout the whole process. In Figure 5 the word decomposer block aims to decompose each English word to a sequence of characters according to the characteristics of English. It can be inferred that the WMBNN model we propose is a universal logic architecture and the word decomposer block acts just like a plug-in which is used to adapted to different kinds of languages.

The experiments in this section will show that WMBNN can also work well for English and we will present the experiment results with optimal parameter settings directly.

English dataset

Amazon reviews. We download some reviews of Amazon about daily necessities (Zhang et al., 2015). These reviews are divided into five categories according to the overall evaluation of the customer and the total number of Amazon reviews we got is about 2,000,000. Then, we randomly

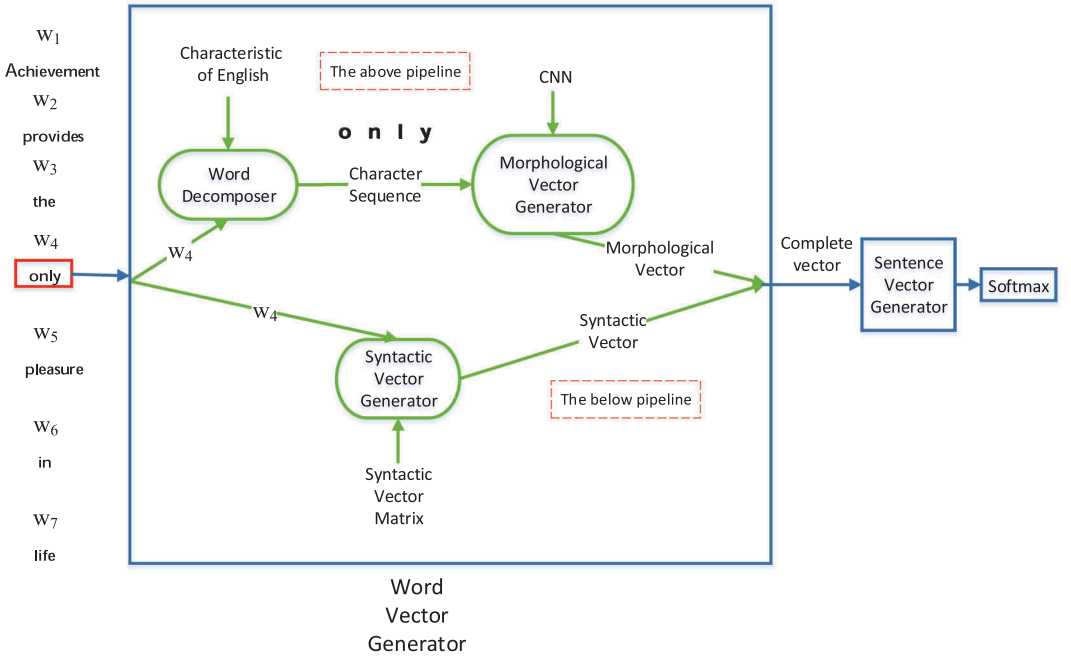

Figure 5. Logic architecture diagram of WMBNN for English.

select some reviews from each class to generate two kinds of dataset, respectively, named Amazon2 dataset and Amazon5 dataset. Amazon2 dataset has two sentiment polarities (100,000 in positive and 100,000 in negative) and Amazon5 dataset has five sentiment polarities (34,000 in very positive, 36,000 in positive, 35,000 in neutral, 37,000 in negative and 37,000 in very negative).

AG news. This dataset is adopted from Zhang et al. (2015). The dataset consists of both articles and descriptions of AG's corpus of news. We choose the 4 largest classes from this corpus to construct our dataset, using only the title and description fields. The number of training samples for each class is 30,000 and testing is 1900.

Movie review. This dataset consists of one sentence per comment on movies. Classification involves detecting positive/negative reviews (Kim, 2014). For this dataset, we randomly split 90% as the training set and the remaining 10% as test set. In this process, we keep a balanced number of items with each label in the training set.

Method of generating complete word vectors for English text

For English text, the Syntactic Matrix is obtained in the same way in Chinese part. As for decomposing English words, Yoon Kim et al. (2016) has proposed one way to decompose English words which is used in our model. As English is a kind of alphabetic language, it is natural to put forward the idea to decompose an English word to a sequence of characters. As there is a blank between every two English words in one sentence, we can generate the matrix C^k directly by stacking every character vector in a word. To guarantee the number of columns is identical in the matrix C^k, we take the length of the longest word throughout the whole corpus as standard to conduct zero paddings which could make sure every word has the same length. Then, CNN operations are employed on the matrix C^k to extract the morphological information of words similar to the way in the Chinese part. Finally, the syntactic word vectors and morphological word vectors are concatenated together to generate the complete word vectors.

Experimental settings

The details of word-building method vary when faced with different languages. An English word can be decomposed into a sequence of characters and the character is the basic element of English. There are total 26 letters in English, so we assign each letter a random vector generated by the distribution of uniform [−0.5, 0.5] with dimension 15. In order to identify the boundary of the words, we generate a random vector with dimension 15 to represent start-of-word and end-of-word character for each word. We also use zero padding to make sure every word embedding matrix has the same length of columns (the same as the longest word on English dataset) and each sentence has the same words (all have the same length as the longest sentence). For alphabets in English, one important option is whether to distinguish upper-case letters from lower-case letters. According to Zhang et al. (2015), it usually (but not always) gives worse results when such a distinction is made. Thus, in our experiment, we convert all upper-case letters to lower-case letters before training the model.

For neural network–based models, the settings in training process (such as regularisation and initialisation) of English text experiment are almost the same as those in Chinese text experiment except for a smaller batch size and different filter widths (3, 4, 5 in English as most root words in English have 3, 4, 5 characters). For the model in Zhang et al. (2015), which also decomposes English words into characters and builds the network upon character sequence, has 6 convolutional layers and 3 fully connected layers. (As this model are proposed for English, so it only appears in comparison experiments for English in this paper and the corresponding Chinese model is called STC that is shown in Table 1) The neural network–based models are all trained by min-batch backpropagation using optimiser RMSprop (Tieleman & Hinton, 2012).The learning rate is set 0.001 at the beginning and reduce by half after each two epoch. The training process is stopped when model validation loss does not decrease anymore.

As for other kinds of comparison models, similar to that in the Chinese experiment part, grid-search approach with 10-fold cross validation are applied to find the best hyper parameters settings. k is set as 10 and cosine similarity is used to measure the similarity in the k-Nearest Neighbor model. For Random Forest model, the number of trees and max-depth are set as 400, 5, respectively, and Gini impurity is used as the split strategy. For IMBNN model, the error correction rate is set as 0.15 and least mean square is used as loss function. For Multinomial Native Bayes, the smoothing parameter is set as 1. The parameter settings of IMBNN and Multinomial Naive Bayes are the same as that in Chinese part. Similar to Chinese part, the above hyperparameters in the comparison models are chosen by grid searching approach. Ten-fold cross-validation strategy is applied in our experiments and the whole dataset is equally divided into 10 parts. We train our model on the training set with enough epochs to obtain the best performance of accuracy on the validation set.

Comparison models and results

Table 4 shows the accuracy of class classification in different models on English dataset. The terms in Table 4 have the same meaning as those in Table 1. CCN refers to the model proposed by Zhang et al. (2015).

From Table 4, we can draw the same conclusion as that in Chinese dataset. Both WMBNN-Glove (94.55% on Amazon2, 55.65% on Amazon5, 89.79% on AG news and 84.15% on Movie review) and WMBNN-W2V (94.32% on Amazon2, 55.77% on Amazon5, 89.93% on AG news and 84.24% on Movie review) outperform the traditional CNN (CNN-Glove and CNN-W2V) and LSTM (LSTM-Glove and LSTM-W2V) model with only syntactic information. At the same time, they both outperform the CCN (92.05% on Amazon2, 53.05% on Amazon5, 86.01% on AG news and 81.26% on Movie review), that is because when doing convolutional operations, characters in two adjacent words will be mixed together which will add noise into the model. It can also demonstrate that the word-vector generated layer in WMBNN (WMBNN-without-syntactic-vectors, WMBNN-W2V, WMBNN-Glove) can effectively extract morphological information from a sequence of characters to represent an English word which is as important as syntactic information for the text classification task. On the other hand, it indicates that our word-building method is suitable for different kinds of

Table 4. Accuracy on different English datasets.

model	Amazon 2	Amazon 5	AG news	Movie review
WMBNN-Glove	94.55%	55.65%	89.79%	84.15%
WMBNN-W2V	94.32%	55.77%	89.93%	84.24%
LSTM-Glove	93.77%	54.56%	87.53%	82.98%
LSTM-W2V	93.61%	54.45%	87.64%	82.73%
CNN-Glove	93.53%	54.26%	86.11%	81.52%
WMBNN-without-syntactic-vectors	93.45%	54.31%	87.53%	82.15%
CNN-W2V	93.47%	54.12%	86.34%	81.71%
CCN	92.05%	53.05%	86.01%	81.26%
IMBNN	91.33%	52.21%	85.44%	80.66%
Bayes	90.84%	51.10%	85.68%	80.15%
Wordterm-Randomforest	77.75%	52.24%	72.95%	77.23%
Wordvec-LR	85.68%	51.24%	61.93%	60.45%
KNN	73.34%	43.37%	64.98%	65.11%

language (Chinese represents one kind of language whose words are a combination of meaning and shape and English is another kind representing the alphabet language which is composed of fixed letters). As for other models, like those in Table 1, Wordvec-LR and Wordterm-Randomforest lose the words sequence information and Bayes as well as IMBNN will behave badly when encountering the sentence containing some negation words or intensity words.

It can be inferred that the WMBNN model we propose is a universal logic architecture and the word decomposer block acts just like a plug-in which is used to adapt to different kinds of languages. So when faced with different kinds of languages, we only need to change the strategy that word decomposer refers to.

Model computational complexity analysis

In this part, we will ananlyse the model computational complexity by dividing it into two stages: training stage and inferring stage. From above experiments, we can see the neural network–based models (including CNNs, LSTMs, WMBNN and CCN) perform much better than other traditional machine learning models on both Chinese and English datasets. Thus, in this part, we only consider neural network–based models.

Training-stage computational complexity analysis

In the training stage, the complexity of the model is proportional to the number of parameters of the model which means that more parameters will lead to more complexity of the model. (X. Li, Qin, Yang, Hu, & Liu, 2016) We then conduct the analysis based on this. In order to simplify the process of calculating the number of parameters in models, the bias parameters in each model are ignored.

Let d_r denote dimension of radicals, c_r denote the number of common radicals in Chinese which is about 400, d_c denote the dimension of characters, c_c denote the number of character which is a fixed number 26. The number of parameters in CNN-W2V/CNN-Glove can be calculated by Eequation (11) in which f_1, f_2, f_3 are the width of filters in CNN, respectively, and n_{CNN} denotes the number of filters. The number of parameters in LSTM-W2V/LSTM-Glove can be calculated by equation (12) in which h_{lstm} denotes the dimension of the hidden state in LSTM. The number of parameters in WMBNN-W2V/WMBNN-Glove can be calculated by Equation (13) in which f_4 and n_3 denote the filter width and filter number in word vector generator, respectively. The number of parameters in WMBNN-without-syntactic-vectors can be calculated by Eequation (14) in which f_5 and f_4 denote the filter width and filter number in word vector generator, respectively. The number of parameters in CCN which consists of six convolutional layers and three fully connected layers can be calculated by Eequation (15) in which n_5, n_6 denote the width of convolution filters and f_5, f_6 denote the number of filters in convolution.

Table 5. Inferring speed of different models.

5Model	Large microblog	Amazon2
WMBNN-W2V/WMBNN-glove	260 records per second	210 records per second
WMBNN-without-syntactic-vectors	258 records per second	209 records per second
CNN	310 records per second	286 records per second
LSTM	262 records per second	211 records per second
CCN	None	122 records per second

$$n_{CNN} * (f_1 * d_{w1} + f_2 * d_{w1} + f_3 * d_{w1}) + 3n_{CNN} * 2 = 264.2K \tag{11}$$

$$4 * h_{lstm} * (h_{lstm} + d_{w1}) + 2 * h_{lstm} = 261.1K \tag{12}$$

$$d_r * c_r + n_3 * (8 * d_r) + 4 * (n_3 + d_{w2}) * (n_3 + d_{w2} + h_{lstm}) = 266.2K \tag{13}$$

$$d_r * c_r + n_4 * (f_4 * d_r) + 4 * (n_4 + h_{lstm}) * n_4 + 2 * h_{lstm} = 264.2K \tag{14}$$

$$2*n_5 * (f_5 * d_c) + 4 * n_5 * (f_6 * d_c) + n_5 * n_6 + n_6 * n_6 + n_6 * n_2 = 322.56K \tag{15}$$

In above formulas, f_1, f_2, f_3, f_4, f_5, f_6 are set as 2, 3, 4, 8, 7, 3, respectively, n_3, n_4, n_5, n_6 are set as 150, 300, 300, 1024, respectively, n_{CNN}, d_{w1} are set as 300, and d_{w2}, h_{lstm} are set as 150 which are the same as those in our above experiments. We can see that these neural network–based models have a similar number of parameters so as to the computational complexity in the training stage.

Inferring stage computational complexity analysis

In inferring stage, the model structure of these neural network models is similar which demonstrates that these neural network models should have the similar inferring stage computational complexity. Then, we conduct some tests to prove it and the results are listed in Table 5.

We conduct tests on Inspur P-8000 with GPU Tesla-k40c and use deep learning framework Keras to implement the models. From Table 5, we can see that the speed of each model are similar indicating that they have similar computational complexity in inferring stage.

Conclusion and future work

In this paper, we introduced WMBNN model for text classification in both Chinese and English datasets. It can extract useful morphological information from a sequence of radicals in Chinese or characters in English. The performance evaluations and model computational complexity analysis show that the proposed WMBNN model with the combination of morphological and syntactic information outperforms the traditional CNN and LSTM models without adding model computational complexity.

The significant contribution in this work is the way of thinking, which presents a new perspective to deal with natural language problems. From the word-building point of view, there are different word building methods in different languages which may contain meaningful structure information. The proposed WMBNN model is a universal architecture to extract structure information from words. Besides, the proposed word decomposer block in the architecture can act as a plug-in module which is adaptable to a different language whose specific decomposing strategy is based on the characteristics of a target language. The conducted experiments have shown that the proposed WMBNN model is effective on both Chinese and English datasets. The model is considered universal since Chinese and English are the most representative language from the word-building point of view, in that, Chinese is a kind of language whose characters are the combination of shape and meaning, and English is an alphabetical language.

As most NLP models use sequence of words as inputs, it would be interesting to further explore how effective the idea of extracting essential morphological information through the radicals in Chinese or the characters in English proposed herein, in other NLP tasks. Furthermore, from the syntax point of view, if more useful information can be extracted from the structure of the whole sentence, not just the sentence as a sequence of the words, it will be a problem worth exploring in the future.

Notes

1. https://keras.io/.
2. http://scikit-learn.org/stable/.
3. https://github.com/guoyuhaoaaa/word-building-method-for-text-classification/.
4. http://www.sina.com.cn/.
5. https://www.taobao.com/.
6. https://www.dianping.com/.

References

Aggarwal, C. C., & Zhai, C. (2012). *Mining text data*. New York, NY: Springer Science & Business Media.

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, *3*(2), 1137–1155.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–166.

Chen, B.-C., Chen, -Y.-Y., Chen, F., & Joshi, D. (2016, February). Business-aware visual concept discovery from social media for multimodal business venue recognition. In *Proceedings of the 30th AAAI conference on artificial intelligence,* (pp. 101–107). Phoenix, Arizona, USA.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, *12*(8), 2493–2537.

Dos Santos, C. N., & Guimarães, V. (2015, July). Boosting named entity recognition with neural character embeddings. In *Proceedings of the fifth named entities workshop,* (pp. 25–33). Beijing China.

Dos Santos, C. N., & Zadrozny, B. (2014, June). Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st international conference on machine learning,* (pp. 1818–1826). Beijing China.

Graves A. (2012). Neural networks. In *Supervised sequence labelling with recurrent neural networks* (pp. 15–35). Berlin, Heidelberg: Springer.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *Computer Science*, *3*(4), 212–223.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.
Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014, June). A convolutional neural network for modelling sentences. In *Proceedings of the 52nd annual meeting of the association for computational linguistics,* (pp. 655–665). Baltimore, MD, USA.
Kim, Y. (2014, October). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 conference on empirical methods in natural language processing,* (pp. 1746–1751). Doha, Qatar.
Kim, Y., Jernite, Y., Sontag, D., & Rush, A. M. (2016, February). Character-aware neural language models. In *Proceedings of the 30th AAAI conference on artificial intelligence,* (pp. 2741–2749). Phoenix, Arizona, USA.
Lai, S., Xu, L., Liu, K., & Zhao, J. (2015, January). Recurrent convolutional neural networks for text classification. In *Proceedings of the 29th AAAI conference on artificial intelligence,* (pp. 2267–2273). Austin, Texas, USA.
LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, *1*(4), 541–551.
LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.
Li, J., Luong, M.-T., Jurafsky, D., & Hovy, E. (2015, September). When are tree structures necessary for deep learning of representations? In *Proceedings of the 2015 conference on empirical methods in natural language processing* (pp. 2304–2314). Lisbon, Portugal.
Li, X., Qin, T., Yang, J., Hu, X., & Liu, T. (2016, December). Lightrnn: Memory and computation- efficient recurrent neural networks. In *Proceedings of Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems, Barcelona, Spain* (pp. 4385–4393).
Liang, C., Paritosh, P., Rajendran, V., & Forbus, K. D. (2016, July). Learning paraphrase identification with structural alignment. In *Proceedings of the 25th international joint conference on artificial intelligence,* (pp. 2859–2865). New York, NY, USA.
Liu, B. (2012). Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, *5*(1), 1–167.
Masdeval, C., & Veloso, A. (2015). Mining citizen emotions to estimate the urgency of urban issues. *Information Systems*, *54*, 147–155.
Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013, December). Distributed representations of words and phrases and their compositionality. In *Proceedings of Advances in neural information processing systems,* (pp. 3111–3119). Lake Tahoe, Nevada, USA.
Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, *2* (1–2), 1–135.
Peng, H., Cambria, E., & Zou, X. (2017, May). Radical-based hierarchical embeddings for chinese sentiment analysis at sentence level. In *Proceedings of the 30th international FLAIRS conference,* (pp. 347–352). Marco Island, Florida, USA.
Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing,* (pp. 1532–1543). Doha, Qatar
Rossi, R. G., Faleiros, T. D. P., Lopes, A. D. A., & Rezende, S. O. (2014). Inductive model generation for text categorization using a bipartite heterogeneous network. *Journal of Computer Science and Technology*, *29*(3), 361–375.
Schulz, A., Mencía, E. L., & Schmidt, B. (2016). A rapid-prototyping framework for extracting small-scale incident-related information in microblogs: Application of multi-label classification on tweets. *Information Systems*, *57*, 88–110.
Sebastiani, F. (2002). Machine learning in automated text categorization. *Acm Computing Surveys*, *34*(1), 1–47.
Shi, X., Zhai, J., Yang, X., Xie, Z., & Liu, C. (2015, July). Radical embedding: Delving deeper to chinese radicals. In *Proceedings of the 53rd annual meeting of the association for computational linguistics* (pp. 594–598). Beijing, China
Sun, J. (2012). *Jieba' Chinese word segmentation tool*. Retrieved from https://github.com/fxsjy/jieba
Sun, Y., Lin, L., Yang, N., Ji, Z., & Wang, X. (2014, November). Radical-enhanced chinese character embedding. In *Neural information processing,* (pp. 279–286). Kuching, Malaysia
Sundermeyer, M., Schlüter, R., & Ney, H. (2012, September). Lstm neural networks for language modeling. In *INTERSPEECH 2012, 13th annual conference of the international speech communication association* (pp. 194–197). Portland, Oregon, USA.
Tang, D., Qin, B., & Liu, T. (2015, September). Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2014 conference on empirical methods in natural language processing,* (pp. 1422–1432). Lisbon, Portugal
Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., & Qin, B. (2014, June). Learning sentiment- specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd annual meeting of the association for computational linguistics,* (pp. 1555–1565). Baltimore, MD, USA.
Tieleman, T., & Hinton, G. (2012). Lecture 6.6-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, *4*, 2.
Wang, Y., Ma, H., Lowe, N., Feldman, M., & Schmitt, C. (2016, February). Business event curation: Merging human and automated approaches. In *Proceedings of the 30th AAAI conference on artificial intelligence,* (pp. 4272–4273). Phoenix, Arizona, USA.
Zhang, J., & Deng, X. (2012, September). *Wubi input system and method. US Patent App. 13/480,323*. Google Patents.

Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In *Advances in neural information processing systems,* (pp. 649–657). Montreal, Quebec, Canada: Curran Associates, Inc.

Zhou, C., Sun, C., Liu, Z., & Lau, F. C. M. (2015). A C-LSTM neural network for text classification. *Computing Research Repository, abs/1511.08630*. Retrieved from http://arxiv.org/abs/1511.08630

Zhu, X., Sobhani, P., & Guo, H. (2015, July). Long short-term memory over recursive structures. In *Proceedings of the 32nd international conference on machine learning,* (pp. 1604–1612). Lille, France.