Moto: Enhancing Embedding with Multiple Joint Factors for Chinese Text Classification

Xunzhu Tang*
National Engineering Research
Center for Big Data Technology
and System, Services Computing
Technology and System Lab,
Cluster and Grid Computing Lab,
School of Computer Science and
Technology, Huazhong University
of Science and Technology
Wuhan, China
tangxz@hust.edu.cn

Rujie Zhu
Department of Electrical
and Computer Engineering
University of Central Florida
Orlando, FL, USA
rujie.zhu@ucf.edu

Tiezhu Sun
Momenta
Suzhou, China
suntiezhu@momenta.ai

Shi Wang*
Institute of Computing
Technology, Chinese Academy
beijing, China
wangshi@ict.ac.cn

Abstract—**Recently, language representation techniques have achieved great performances in text classification. However, most existing representation models are specifically designed for English materials, which may fail in Chinese because of the huge difference between these two languages. Actually, few existing methods for Chinese text classification process texts at a single level. However, as a special kind of hieroglyphics, radicals of Chinese characters are good semantic carriers. In addition, Pinyin codes carry the semantic of tones, and Wubi reflects the stroke structure information, *etc*. Unfortunately, previous researches neglected to find an effective way to distill the useful parts of these four factors and to fuse them. In our works, we propose a novel model called Moto: Enhancing Embedding with Multiple Joint Factors. Specifically, we design an attention mechanism to distill the useful parts by fusing the four-level information above more effectively. We conduct extensive experiments on four popular tasks. The empirical results show that our Moto achieves SOTA 0.8316 (F_1-score, 2.11% improvement) on Chinese news titles, 96.38 (1.24% improvement) on Fudan Corpus and 0.9633 (3.26% improvement) on THUCNews.**

I. Introduction

Different from the English-like languages whose words can be spelt out according to their pronunciation and meanings are related with words themselves, semantics of Chinese are relevant to characters and highly associated with component parts (*i.e.,* radicals [1], [2]), structure of characters (*i.e.,* Wubi codes [3]), and tones of pronunciation (*i.e.,* Pinyin codes [4]). Over the past years, many works applied only one of different aspects of Chinese characters on sequence-to-sequence model to enhance the ability of capturing semantic features. We advocate fusing all the four aspects of Chinese characters (*i.e.,* multiple joint factors) to enhance Chinese Embedding, which would bring much better performance for Chinese texts classification [5].

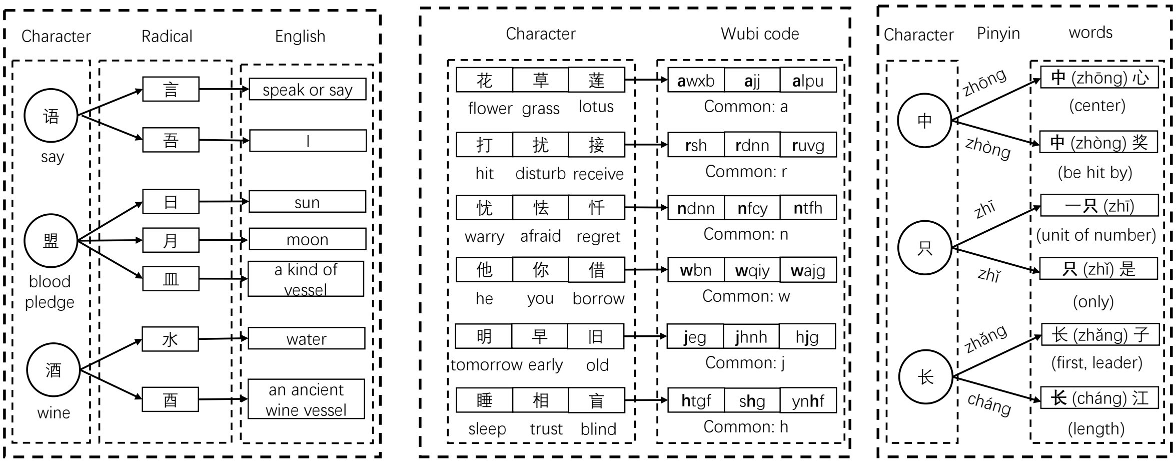
As a kind of pictograph language, the uniqueness of Chinese is that its character system is based on hieroglyphics, which means that Chinese characters have their raw meanings. In other words, not only Chinese characters themselves can express specific meanings, but also their component parts are important carriers of semantics, which is the main point where Chinese differ from English. As shown in Figure 1(a), ‘盟’ (blood pledge) consists three radicals : ‘日 (sun)’, ‘月’ (moon), and ‘皿’ (a kind of vessel). Furthermore, the positions of these radicals in characters are also significant in hieroglyphics. For example, as shown in Figure 1(b), ‘花’ (flower), ‘草’ (grass), and ‘莲’ (lotus) have one common radical ‘艹’, the same structure (upper-down), and the same position (*i.e.,* ‘艹’ is in the upper position, which was recorded as ’*a*’ in Wubi codes), which means they are all plants. It is not difficult to see that radicals and Wubi codes could help us to recognize semantics for classifying Chinese texts. In addition, Pinyin codes could also help us to capture the semantic of tone, as shown in Figure 1(c).

Additionally, there has been a lot of works aiming to employ Wubi codes on Chinese Word Segmentation (CWS) [3], radicals on Chinese text classification [1], [2], and Pinyin codes on Chinese embeddings [4]. However, these works simply use single aspect of characters like radicals to enhance character-level embeddings.

Inspired by the importance of radicals of characters, Wubi codes, and Pinyin codes, we conduct an explorative study in Chinese text classification with attention mechanism to jointly leverage four granularities of features, which we call Moto in this paper. The main contributions are threefold:

- We first employ the attention mechanism to capture the effective parts four-granularity features (*i.e.,* characters, radicals, Wubi codes, and Pinyin codes.)
- We first design a novel mechanism to confirm the weights among these four-granularity features dynamically.
- We conduct extensive experiments on four real-world and public datasets in four granularities respectively, and demonstrate the effectiveness of characters, radicals, Wubi codes, and Pinyin codes.

*Joint first authors



(a) Radical (b) Wubi (c) Pinyin

Fig. 1: We employ three kind of representations to enhance the character embedding. Figure (a) indicates that radicals can show more details of characters; Figure (b) shows that Wubi code can capture the structure information of characters; And figure (c) expresses that the Pinyin (with pronounce) is important to Chinese characters.

II. Multiple Embeddings

In this section, we will discuss the priority of employing four different granularities, including characters, radicals, Wubi codes, and Pinyin codes, as shown in Figure 2.

Character is usually recognized as the smallest unit to process Chinese text classification. Recent works show that character embeddings are the most fundamental inputs for neural networks [3], [6]–[8] since it is easy to learn contextual information with sequence-to-sequence models. However, Chinese character system is based on hieroglyphics, whose component parts of characters are also the carriers of semantics.

Radical or radical-like components serving as the basic units for building Chinese characters has been explored in [1], [9]. Commonly, radicals has the following two features. The first one is that one radical normally has one or two types. '言' (speak) is itself, but it becomes '讠' in character '语'. The second is that radicals have specific meanings. In Figure 1(a), the character '语' (*talk or speak*) have radicals : '讠' (*the same as 言, speak*) and '吾' (*I*). Obviously, radical provides extra photographic features of characters.

Wubi is another effective representation of Chinese characters, which includes more comprehensive structure information compared to radical. Each element in a Wubi code represents a type of structure (or stroke) in characters. In Figure 1(b), '花' (flower), '草' (grass), and '莲' (lotus) are all related to plants, and their Wubi codes 'awxb', 'ajj', and 'alpu' have one common letter '*a*', which is corresponding to radical '艹'. Therefore, Wubi is an efficient approach to capture structure features of Chinese characters.

Pinyin is a English-like expression approach of Chinese characters. Besides, Pinyin is highly relevant to semantics - one character may have multiple pronunciations corresponding to different semantic meanings [3], which is called polyphone in Chinese. Figure 1(c) shows several polyphone characters. '中' has two pronunciations (Pinyin). When pronounced as 'zhōng' in '中心', it means 'center'. However, when it refer to the meaning of 'be hit by' when pronounced in '中奖'. Obviously, it is beneficial for Chinese characters to use Pinyin code to capture phonetic information.

III. Details of Moto Model

In this section, we introduce our joint enhanced character embedding model (Moto), which utilizes radical, Wubi code, and Pinyin as supplementary input text. The challenge is that how to distill the important parts of these factors and how to conform the weights among them. The paper extends the method of attention mechanism [10] to infer the weights, which will be discussed in III-C. As shown in Figure 2, Moto mainly contains four parts: *Input Layer*, *Bidirectional LSTM Layer*, *Attention Layer*, and *Prediction Layer*.

A. Input Layer

Given a Chinese-character sequence C which contains lc characters, *i.e.*, $C = \{c_1, c_2, \ldots, c_{lc}\}$, where each character c_i ($1 \leq i \leq lc$ *(length of characters)*) is an independent item in C. Meanwhile, C will be mapped into radicals, Wubi, and Pinyin respectively by the usage of *Open Chinese dictionary* [1], Wubi Library [2], and Pypinyin Library [3], *i.e.*, lr (the length of radicals) radical-level radicals R=$\{r_1, r_2, \ldots, r_{lr}\}$, lw (the length of Wubi codes) Wubi codes W=$\{w_1, w_2, \ldots, w_{lw}\}$, and lp (the length of Pinyin codes) Pinyin codes Py=$\{py_1, py_2, \ldots, py_{lp}\}$. Then we retrieve four granularities of features (*i.e., C, R, W, Py*) and obtain four embedding matrices using word2vec tool [4] [11]. As shown in Figure 2, the embedding of sequence C

[1] http://www.kaifangcidian.com/han/chaizi

[2] https://github.com/sfyc23/python-wubi

[3] https://pypi.org/project/pypinyin/

[4] https://radimrehurek.com/gensim/

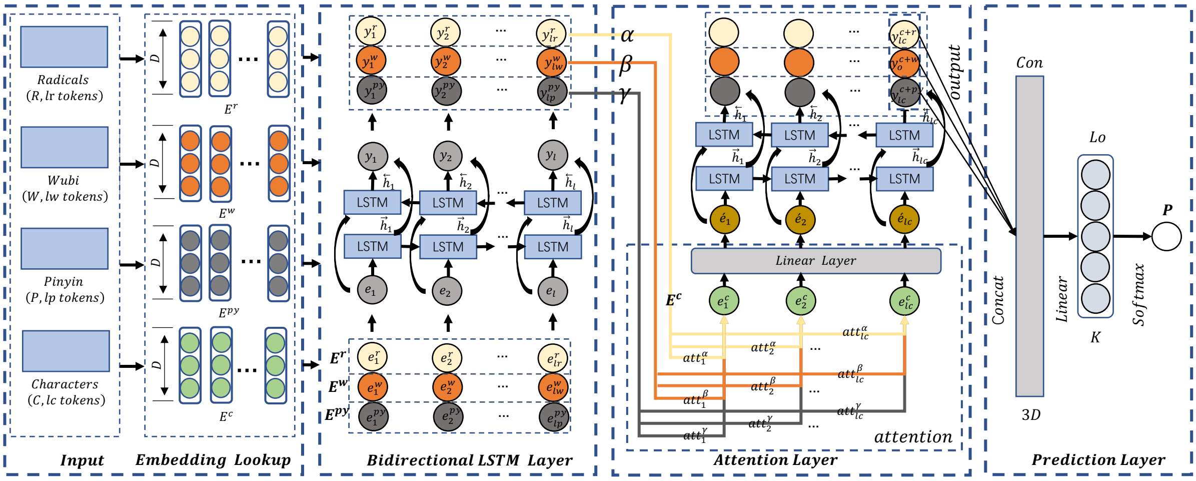


Fig. 2: Network architecture of **Moto**, including four-granularity representations of Chinese: Radicals, Wubi, Pinyin, and Characters.

can be represented of E^c= $\{e_1^c, e_2^c, \ldots, e_{lc}^c,\}$ (where e_i^c is the representation of c_i). Similarly, E^r= $\{e_1^r, e_2^r, \ldots, e_{lr}^r,\}$ (where e_i^r is the representation of r_i), E^w= $\{e_1^w, e_2^w, \ldots, e_{lw}^w,\}$ (where e_i^c is the representation of c_i), E^{py}= $\{e_1^{py}, e_2^{py}, \ldots, e_{lp}^{py},\}$ (where e_i^{py} is the representation of py_i). For simplicity, we set the vector dimension of each level embeddings as D, which means $E^c \in R^{lc\times D}$, $E^r \in R^{lr\times D}$, $E^w \in R^{lw\times D}$, $E^{py} \in R^{lp\times D}$. Then we feed them in BiLSTM layer directly.

B. Bidirectional LSTM Layer

In this section, we employ BiLSTM to capture contextual information of input sequence and obtain independent-contextual representation. LSTM [12] is an advanced version of recurrent neural network (RNN) with extra forget and memory which are employed to alleviate the gradient vanishing problem and keep the term information as long as possible. Given a specific feature embedding sequence E= $\{e_1, e_2, \ldots, e_n\}$, the whole progress in the forward LSTM is calculated as follows:

$$\begin{bmatrix} \overrightarrow{i_t} \\ \overrightarrow{f_t} \\ \overrightarrow{o_t} \\ \overrightarrow{\widetilde{c}_t} \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \left(W^T \begin{bmatrix} e_t \\ \overrightarrow{h_{t-1}} \end{bmatrix} + b \right) \tag{1}$$

$$\begin{aligned} \overrightarrow{c_t} &= f_t * \overrightarrow{c_{t-1}} + \overrightarrow{i_t} * \overrightarrow{\widetilde{c}_t} \\ \overrightarrow{h_t} &= \overrightarrow{o_t} * \tanh \overrightarrow{c_t} \end{aligned} \tag{2}$$

where i_t, f_t, o_t, and $\widetilde{c}_t$ denote a set of input, forget, output, and new layer to update current information c_t respectively. Moreover, W equals to the concatenation of W_i (a matrix parameter in input gate), W_f (a matrix parameter in forget gate), W_o (a matrix parameter in output gate), and W_c (a matrix parameter in *tanh* layer). The progress above can be described as equation $W = W_i \oplus W_f \oplus W_o \oplus W_c$, where symbol $\oplus$ represents the concatenation function. Similar to W, b= $b_i \oplus b_f \oplus b_o \oplus b_c$. In addition, symbol $\sigma(\cdot)$ indicates the sigmoid function. Similar to forward LSTM, the hidden state of t-th step in the backward LSTM can be represented as $\overleftarrow{h_t}$. Then we concatenate $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ as y_t, which is the hidden output of each BiLSTM cell at the t-th step, and the process is computed by $y_t = \overrightarrow{h_t} \oplus \overleftarrow{h_t}$.

As shown in Figure 2, there are three input embeddings in the Bidirectional LSTM Layer (*i.e.*, E^r, E^w, and E^{py}), which will be fed to three different BiLSTM network (*i.e.*, $BiLSTM^r$, $BiLSTM^w$, and $BiLSTM^{py}$ which share parameters). Then we can get three related outputs from these BiLSTM networks, *i.e.*, Y^r=$\{y_1^r, y_2^r, \ldots, y_{lr}^r\}$, Y^w=$\{y_1^w, y_2^w, \ldots, y_{lr}^w\}$ and Y^{py}=$\{y_1^{py}, y_2^{py}, \ldots, y_{lr}^{py}\}$, which will be taken into calculation in Section III-C.

C. Attention Layer

BiLSTMs provide three outputs of Y^r, Y^w, and Y^{py} in the last section. For a specific output Y^r, different element $y_i^r \in Y^r$ (where $1 \leq i \leq lr$) has different effect on Chinese character-level semantics. In this section, we design an attention mechanism to calculate the weight of different element y_i^r in Y^r at the j-th $BiLSTM^c$ cell. The $\boldsymbol{AttentionLayer}$ in Figure 2 shows that each time of character-level BiLSTM network has one attention input, *i.e.*, $att_i^\alpha \in att^\alpha$= $\{att_1^\alpha$, att_2^α, $\ldots$, $att_{lc}^\alpha\}$, $att_i^\beta \in att^\beta$= $\{att_1^\beta$, att_2^β, $\ldots$, $att_{lc}^\beta\}$, and $att_i^\gamma \in att^\gamma$= $\{att_1^\gamma$, att_2^γ, $\ldots$, $att_{lc}^\gamma\}$.

For ease of exposition, we take att^α as the main example in the next phase of inference. The sequence network to deal with character-level embeddings $\boldsymbol{E^c}$ is donated as $BiLSTM^c$. In each time $BiLSTM^c$ receives a vector of embedding of $y_j^c \in Y^c$= $\{y_1^c$, y_2^c, $\ldots$, $y_{lc}^c\}$, where y_j^c= $\overrightarrow{h_j^c} \oplus \overleftarrow{h_j^c}$.

In order to get weight of element y_i^r in Y^r at j-th time of $BiLSTM^c$, we utilize dot function to figure out relevance between y_i^r and y_{j+1}^c. We denote the relevance between y_i^r and y_j^c as $re_{i,j}$, and the process of calculation is as follows:

$$\begin{aligned} re_{i,j} &= y_i^{rT} y_j^c \\ re_{:,j} &= \{re_{1,j}, re_{2,j}, \ldots, re_{lr,j}\} \end{aligned} \tag{3}$$

Then we employ *Softmax* to normalize $\boldsymbol{re}_{:,j}$ to get the attention distribution α=$\{\alpha_{1,j},\ \alpha_{2,j},\ \ldots,\ \alpha_{lr,j}\}$, where $\alpha_{i,j}$ is calculated as Equation 4.

$$\alpha_{i,j} = \frac{\exp\left(re_{i,j}\right)}{\sum_{k=1}^{lr} \exp\left(re_{k,j}\right)}, where \sum_{i=1}^{lr} \alpha_i = 1 \tag{4}$$

As a result, all weights of $\{y_1^r,\ y_1^r,\ \ldots,\ y_{lr}^r\}$ have been figured out. Here we use weighted arithmetic to value the whole effect of radical-level output Y^r on character-level embedding. The calculation of weighted value and concatenation are as follows:

$$\begin{aligned} att_j^{\alpha} &= \sum_{i=1}^{lr} \alpha_{i,j} \times y_i^r \\ \acute{e}_{j+1}^c &= Linear(att_j^{\alpha} \oplus e_{j+1}^c) \end{aligned} \tag{5}$$

where att_j^{α} is the weighted effect of radical-level embedding in *j*-th output of $BiLSTM^c$ cell.

After the attention operation, the corresponding attention of radical-level embeddings to each origin input e_j^c is available, *i.e.,* att^{α}= $\{att_1^{\alpha},\ att_2^{\alpha},\ \ldots,\ att_{lc}^{\alpha}\}$. Instead of input e_{j+1}^c, we feed the new input $\acute{e}_{j+1}^c$ into the neural network, where $\acute{e}_{j+1}^c$ $\in$ $\acute{E}^c$= $\{\acute{e}_1^c,\ \acute{e}_2^c,\ \ldots,\ \acute{e}_{lc}^c\}$. After decades of epochs of training $BiLSTM^c$ network with input $\acute{E}^c$, it outputs a contextual sequence Y^{c+r}= $\{y_1^{c+r},\ y_2^{c+r},\ \ldots,\ y_{lc}^{c+r}\}$.

Similar to the attention operation of radical-level embedding on characters, Moto figures out that the output of fusion information of Wubi and Pinyin on characters respectively, *i.e.,* Y^{c+w}= $\{y_1^{c+w},\ y_2^{c+r},\ \ldots,\ y_{lc}^{c+w}\}$ and Y^{c+py}= $\{y_1^{c+py},\ y_2^{c+py},\ \ldots,\ y_{lc}^{c+py}\}$.

D. Prediction Layer

Last but not least, we employ the last items of Y^{c+py}, Y^{c+py} and Y^{c+py} (*i.e.,* y_{lc}^{c+py}, y_{lc}^{c+py} and y_{lc}^{c+py}) as the final output, then we conduct a concatenation operation on them and get a comprehensive representation *Con* $\in$ R^{3D} (*i.e., Con*= $y_{lc}^{c+r+w+py}$ = y_{lc}^{c+r} $\oplus$ y_{lc}^{c+w} $\oplus$ y_{lc}^{c+py}). After that, we feed *Con* into a fully-connected neural network to obtain an output vector *Lo* $\in$ R^K (*Lo*= $\{Lo_1,\ Lo_2,\ \ldots,\ Lo_K\}$, and *K* is the number of classes in classification task), the whole calculation is shown as follows:

$$Lo = \sigma(Con \times W) \tag{6}$$

where $W \in R^{3D \times K}$ is the weight matrix for dimension transformation, and $\sigma(\cdot)$ is an activation function named *sigmoid.*

Finally, we utilize a *Softmax* layer to map each element in *Lo* to a conditional probability. The calculation of probability distribution and class index corresponding to the max one are shown as follows:

$$\begin{aligned} p^{Lo_i} &= \frac{\exp\left(Lo_i\right)}{\sum_{k=1}^{K} \exp\left(Lo_i\right)} \\ P &= \arg\max(P^{Lo}) \end{aligned} \tag{7}$$

where $\sum_{i=1}^{K} p^{Lo_i}$= 1 and P^{Lo}= $\{p^{Lo_1}, p^{Lo_1},\ \ldots,\ p^{Lo_K}\}$.

E. Model Training

Since what we are trying to solve is a multi-class classification task, we apply the cross-entropy [2], [13] loss function to train our model Moto, and the goal is to minimize the following *Loss*:

$$Loss = -\sum_{C \in Corpus} \sum_{i=1}^{K} p_i(C) \log p_i(C) \tag{8}$$

where *C* is the input character-level input text. *Corpus* denotes the training corpus and *K* is the number of classes. In the period of training, we utilize *Adam* [14] as optimizer to update the parameters of Moto. In addition, all *BiLSTMs* share properties, including weights and biases.

IV. Experiments

A. Datasets

To make a objectively comparison with baseline models, we conduct experiments separately on four datasets with gold standard classification labels. These four datasets include Chinese news titles (#1 and #2) [5], Fudan corpus [6] , and THUCNews [7].

Dataset#1 The dataset of Chinese news titles contains 47,952 titles with 32 classes (*i.e.,* the topics of news) for training and 15,986 titles for testing. In order to preserve the justice of the comporison with [2] and [13], we do not filter out any texts.

Dataset#2 To test the difference among these four aspects, we need to keep the purity of the dataset. To do so, we filter the original dataset*#1* by removing the texts whose non-Chinese ratio is larger that 20%, which refers to the approach proposed in [2]. The processed texts is remarked as dataset*#2*.

Dataset#3 The dataset of Fudan corpus is a public dataset for Chinese text classification task. In this paper, we take 13,649 for training, and the remaining 4,549 for testing.

Dataset#4 The dataset of THUCNews contains 836,036 titles with 14 classes, 627,027 of which for training and the other 209,009 for testing.

B. Experimental Setup

Initial Setting. We use *Open Chinese dictionary*, *Wubi library*, and *Pypinyin Library* to transform the character texts to radical, Wubi, and Pinyin texts, respectively, as discussed in Section III-A. Moreover, we take the former average length *lavg* tokens into computing. And if the length of a sentence is smaller than *lvag*, we will use character ‘一’ to make up the sentence until the length of it equals *lvag*.

Embedding Setting. Since the performance of deep learning models is highly related to the quality of input embeddings, we utilize the public word2vec tool (*Gensim*) to train embeddings for characters, radicals, Wubi codes, and Pinyin codes based on the large corpora [8]. The dimension of those embeddings are all set to 256 (*i.e., D*= 256). In addition, since the average length of texts in Fudan corpus and THUCNews

[5] https://pan.baidu.com/s/1mgBTFOO

[6] https://github.com/yzwww2019/Fudan-corpus

[7] http://thuctc.thunlp.org/message

[8] https://spaces.ac.cn/archives/4304

TABLE I: Experimental results of different methods on Chinese news titles, Fudan Corpus, Douban movie review, and THUCNews.

Methods	Chinese news titles dataset #1	Chinese news titles dataset #2	Fudan Corpus	THUCNews
	F1(P,R)	F1(P,R)	F1(P,R)	F1(P,R)
SVM+BOW(C)	0.7421 (0.7440, 0.7420)	0.7252 (0.7268, 0.7255)	0.8434 (0.8373, 0.8495)	0.8713 (0.8811, 0.8618)
SVM+BOW(R)	0.4697 (0.4652, 0.4809)	0.4691 (0.4636, 0.4813)	0.8187 (0.8216, 0.8158)	0.8641 (0.8637, 0.8646)
SVM+BOW(W)	0.6021 (0.6041, 0.6002)	0.4852 (0.4783, 0.4923)	0.8303 (0.8229, 0.8378)	0.8638 (0.8597, 0.8679)
SVM+BOW(Py)	0.7290 (0.7309, 0.7271)	0.6702 (0.6874, 0.6539)	0.8359 (0.8367, 0.8352)	0.8703 (0.8778, 0.8629)
Four LSTMs (C + R + W + Py)	0.8072 (0.8078, 0.8074)	0.7904 (0.7912, 0.7910)	0.8826 (0.8841, 0.8811)	0.9018 (0.9022, 0.9014)
Four BiLSTMs (C + R + W + Py)	0.8098 (0.8103, 0.8103)	0.7915 (0.7925, 0.7921)	0.8899 (0.8990, 0.8809)	0.9122 (0.9191, 0.9054)
RAFG	0.8181 (0.8181, 0.8187)	0.7999 (0.7993, 0.8010)	0.9172 (0.9201, 0.9144)	0.9002 (0.9033, 0.8972)
cw2vec(stroke-level)	– (–, –)	– (–, –)	0.9520 (0.9528, 0.9511)	0.9329 (0.9433, 0.9227)
C-LSTMs (C)	0.8108 (0.8102, 0.8114)	0.7931 (0.7944, 0.7929)	0.8801 (0.8828, 0.8774)	0.9033 (0.9054, 0.9012)
C-LSTMs (C + R + W + Py)	0.8163 (0.8177, 0.8149)	0.7956 (0.7951, 0.7972)	0.8823 (0.8775, 0.8871)	0.9036 (0.9068, 0.9004)
C-BiLSTMs (C)	0.8140 (0.8153, 0.8127)	0.7757 (0.7754, 0.7922)	0.9213 (0.9309, 0.9118)	0.9236 (0.9290, 0.9183)
C-BiLSTMs (C + R + W + Py)	0.8211 (0.8246, 0.8177)	0.7939 (0.7957, 0.7922)	0.9264 (0.9384, 0.9147)	0.9294 (0.9332, 0.9257)
Moto(BiLSTM)	**0.8316** (0.8346, 0.8287)	**0.8168** (0.8192, 0.8144)	**0.9638** (0.9671, 0.9605)	**0.9633** (0.9679, 0.9588)

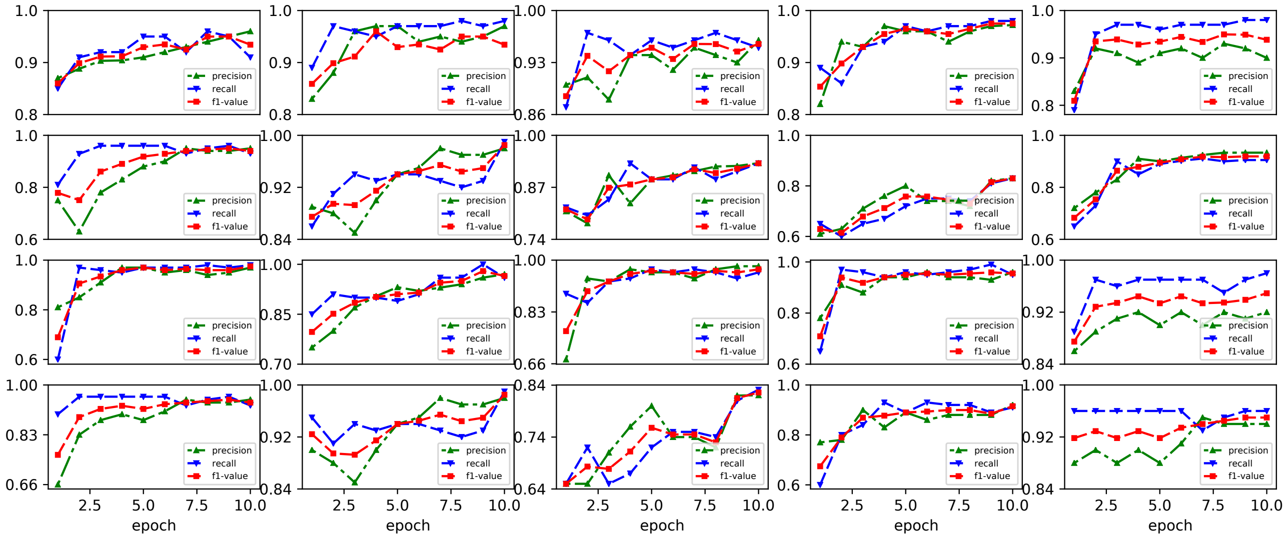


Fig. 3: Details of the validations on the dataset of Fudan corpus, in which there are 20 classes, and xlabel refers to the number of epochs.

is too longer for LSTMs to deal with, we take 1D-CNN [15] to convolute the texts in token dimension, (*i.e.,* the dimension of embedding is still kept at 256, but the token number is convoluted to). For example, the original embedding matrix shape 32 × 4058 × 256 will be transformed to 32 × 18 × 256 as the input matrix shape. Finally, we conduct our experiment on 2 pieces of P100 GPU.

Training Setting. According to the previous training experience, we set the the dimension of hidden vectors in BiLSTM to 256, and set dropout rate to 50% to escape overfitting. Moreover, the learning rate is set to 0.001 and we take Adam [14] as optimizer for gradient descent computing. Furthermore, we set batch size to 32 empirically, and employ *Precision (P), Recall (R), and* F_1*-value* to evaluate the performance [16], [17], which is computed as follows:

$$F_1 = \frac{2 \times (Precision \times Recall)}{Precision + Recall} \tag{9}$$

C. Baseline Methods

We compare our model with following baseline models in Chinese short text classification.

- SVM+BOW. To evaluate the performance of radicals, Wubi codes, and Pinyin codes, we utilize tf-idf weights of characters (*C*), radicals (*R*), Wubi codes (*W*), and Pinyin codes (*Py*) as features separately, and train SVM classifier with *liblinear* [9].
- Four LSTMs/BiLSTMs. We employ four LSTMs to process *C, R, W, and Py* as a whole baseline model, whose four corresponding hidden output would be concatenated into a vector. Similar to four LSTMs, we utilize four BiLSTMs as another baseline to test the effectiveness of bidirectional setting.
- RAFG [2]. RAFG is a four-granularity (*i.e.,* characters, radicals, character words, and radical words) model based on attention mechanism.

[9] https://www.csie.ntu.edu.xn–tw/cjlin/liblinear/-784l

- cw2vec [18]. cw2vec is a method for learning Chinese word embeddings in stroke-level information based on n-grams algorithm.
- C-LSTMs / C-BiLSTMs [13]. C-LSTMs employs two independent LSTMs to capture word and character features, which would be concatenated together. C-BiLSTMs is the bidirectional version of C-LSTMs.

D. Experimental Results

Table I demonstrates the F_1*-value*, Precision, and Recall of these baseline models and our Moto. In the following, we introduce these results in detail.

We provide the comparison results with SVM+BOW employing characters, radicals, Wubi codes, and Pinyin codes as features respectively. Table I shows that SVM + BOW (C) achieves the best average F_1-value 0.7955, 2.5% higher than SVM + BOW (Py) in four Chinese text classification tasks. At the same time, Wubi gets average F_1-value 0.6954, as well radical gets 0.6554. The results indicate that all these four aspects are carriers of semantics in Chinese, and character plays the most important role in them.

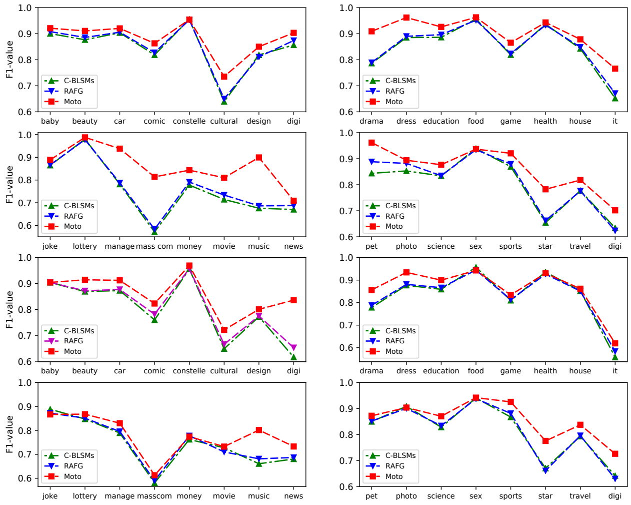


Fig. 4: Detailed comparison on the dataset of Chinese news titles, Sub-figures in former two rows describe the dataset#1, and sub-figures in the later two rows are related to dataset#2.

When comparing four LSTMs (C + R + W + Py), Four BiLSTMs (C + R + W + Py), RAFG, and cw2vec, we can find that RAFG which takes attention mechanism achieves the best performance, whose average F_1-value is 0.8589, higher than Four LSTMs (0.8455)) and Four BiLSTMs (0.8509). Moreover, cw2vec achieves the best performance in Fudan Corpus and THUCNews. Additionally, for C-LSTMs (C), C-LSTMs (C + R + W + Py), C-BiLSTMs(C), and C-BiLSTMs(C + R + W + P), the results indicate that methods with bidirectional version achieve better performance. At the same time, four-granularity model is better than single character-level model. Figure 4 plots that the comparison in F_1-value among C-BiLSTMs, RAFG, and our model Moto. We can see that Moto achieves the best performance in the most classes in dataset #1 and dataset #2.

For Moto, as shown in Figure 2, we also employ four-granularity facts (*i.e,* but different from RAFG) based on attention mechanism. We conduct Moto on four datasets mentioned above. As a result, we can see that they achieve better score in F_1*-value*, *Precision*, and *Recall* compared with SVM+BOW, four LSTMs/BiLSTMs, RAFG, and cw2vec on datasets shown in Table I. In addition, Moto gets 0.9638 of F_1-value 1.24% which is higher than the second method cw2vec in Fudan Corpus, and the detailed comparison in Fudan Corpus is shown in Figure 3. Finally, we conduct Moto in different aspact of these four granularities, and we can order the effectiveness of them as C $>$ Py $>$ R $>$ W.

V. Conclusion

We propose a novel method combining four granularities (*i.e., characters, radicals, Wubi codes, and Pinyin codes*) based on attention mechanism. Through the experiments on these four aspects, the order of importance in semantic of Chinese is that Moto (C) $>$ Moto (Py) $>$ Moto (R) $>$ Moto (W). In addition, the results in group Moto (C+X) (*X= R, W, or Py*) demonstrate that characters, radicals, Wubi codes, and Pinyin codes are unquestionably important semantic features in Chinese text classification. Our method Moto is 3.02% higher the second method C-BiLSTM (C + R + W + Py) in average F_1-value in four datasets. Specifically, Moto improve the F_1-scores of four datasets: 1.28%, 2.11%, 1.24%, and 3.26%. In addition, our Moto achieves the SOTA in precision (1.89% average improvement).

References

[1] X. Shi, J. Zhai, X. Yang, Z. Xie, and C. Liu, "Radical embedding: Delving deeper to chinese radicals," in *Proceedings of ACL*, 2015, pp. 594–598.

[2] H. Tao, S. Tong, H. Zhao, T. Xu, B. Jin, and Q. Liu, "A radical-aware attention-based model for chinese text classification," in *AAAI*, 2019.

[3] J. Zhou, J. Wang, and G. Liu, "Multiple character embeddings for chinese word segmentation," in *Proceedings of ACL*, 2019, pp. 210–216.

[4] S. Chen, H. Zhao, and R. Wang, "Neural network language model for chinese pinyin input method engine," in *Proceedings of PACLIC*, 2015, pp. 455–461.

[5] F. Peng, X. Huang, D. Schuurmans, and S. Wang, "Text classification in asian languages without word segmentation," in *Proceedings of IRAL*. Association for Computational Linguistics, 2003, pp. 41–48.

[6] X. Chen, X. Qiu, C. Zhu, P. Liu, and X. Huang, "Long short-term memory neural networks for chinese word segmentation," in *Proceedings of EMNLP*, 2015, pp. 1197–1206.

[7] D. Cai and H. Zhao, "Neural word segmentation learning for chinese," in *Proceedings of ACL*, 2016, pp. 409–420.

[8] D. Cai, H. Zhao, Z. Zhang, Y. Xin, Y. Wu, and F. Huang, "Fast and accurate neural word segmentation for chinese," in *Proceedings of ACL*, 2017, pp. 608–615.

[9] Y. Li, W. Li, F. Sun, and S. Li, "Component-enhanced chinese character embeddings," in *Proceedings of EMNLP*, 2015, pp. 829–834.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[11] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of NIPS*, 2013, pp. 3111–3119.

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[13] Y. Zhou, B. Xu, J. Xu, L. Yang, and C. Li, "Compositional recurrent neural networks for chinese short text classification." IEEE, 2016, pp. 137–144.

[14] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[15] Y. Kim, “Convolutional neural networks for sentence classification,” in *Proceedings of EMNLP*, 2014, pp. 1746–1751.

[16] A. Hotho, A. Nürnberger, and G. Paaß, “A brief survey of text mining.” in *Ldv Forum*, vol. 20, no. 1. Citeseer, 2005, pp. 19–62.

[17] L. Qiao, H. Zhao, X. Huang, K. Li, and E. Chen, “A structure-enriched neural network for network embedding,” *Expert Systems with Applications*, vol. 117, pp. 300–311, 2019.

[18] S. Cao, W. Lu, J. Zhou, and X. Li, “cw2vec: Learning chinese word embeddings with stroke n-gram information,” in *AAAI*, 2018.