Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Shared-weight multimodal translation model for recognizing Chinese variant characters*

Yuankang SUN[†1,2], Bing LI[†1,2], Lexiang LI[†1,2], Peng YANG[†‡1,2], Dongmei YANG[3]

[1]*School of Computer Science and Engineering, Southeast University, Nanjing 210000, China*

[2]*Key Laboratory of Computer Network and Information Integration, Ministry of Education, Southeast University, Nanjing 210000, China*

[3]*School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China*

[†]E-mail: syk@seu.edu.cn; libing@seu.edu.cn; lexiangli@seu.edu.cn; pengyang@seu.edu.cn

Received June 11, 2024; Revision accepted Oct. 10, 2024; Crosschecked June 16, 2025

Abstract: The task of recognizing Chinese variant characters aims to address the challenges of semantic ambiguity and confusion, which potentially cause risks to the security of Web content and complicate the governance of sensitive words. Most existing approaches predominantly prioritize the acquisition of contextual knowledge from Chinese corpora and vocabularies during pretraining, often overlooking the inherent phonological and morphological characteristics of the Chinese language. To address these issues, we propose a shared-weight multimodal translation model (SMTM) based on multimodal information of Chinese characters, which integrates the phonology of Pinyin and the morphology of fonts into each Chinese character token to learn the deeper semantics of variant text. Specifically, we encode the Pinyin features of Chinese characters using the embedding layer, and the font features of Chinese characters are extracted based on convolutional neural networks directly. Considering the multimodal similarity between the source and target sentences of the Chinese variant-character-recognition task, we design the shared-weight embedding mechanism to generate target sentences using the heuristic information from the source sentences in the training process. The simulation results show that our proposed SMTM achieves remarkable performance of 89.550% and 79.480% on bilingual evaluation understudy (BLEU) and F1 metrics respectively, with significant improvement compared with state-of-the-art baseline models.

Key words: Chinese variant characters; Multimodal model; Translation model; Phonology and morphology
https://doi.org/10.1631/FITEE.2400504 **CLC number:** TP391

1 Introduction

To evade platform monitoring, a substantial volume of sensitive textual content is widely disseminated across social media platforms. These malicious substitutions of sensitive text often use variations such as morphological similarities, homophones, and traditional characters. The 2021 China Computer Federation (CCF) Big Data and Computing Intelligence Competition (CCF BDCI) dataset (https://www.datafountain.cn/competitions/508/datasets) is presented in Table 1. Variant characters can be broadly categorized into four categories: phonological variants, morphological variants, symbolic variants, and syntactic variants, which are further divided into nine specific patterns.

The models for the Chinese variant-character-recognition task are mainly divided into two

‡ Corresponding author

* Project supported by the Consulting Project of the Chinese Academy of Engineering (No. 2023-XY-09), the National Natural Science Foundation of China (No. 62272100), and the Academy–Locality Cooperation Project of the Chinese Academy of Engineering (No. JS2021ZT05)

ORCID: Yuankang SUN, https://orcid.org/0000-0003-3217-020X; Peng YANG, https://orcid.org/0000-0002-1184-8117

Table 1 Nine common malicious variants of Chinese characters

Variant category	Variant mode	Text
Normal	–	刷单加微信
Phonological variants	Homophone	刷单加威信
	Pinyin substitution	刷单加wei信
	Pinyin abbreviation	刷单加wx
Morphological variants	Traditional character substitution	刷單加微信
	Similar character substitution	刷单加徽信
	Radical split	刷单力口微信
Symbolic variants	Special symbol substitution	刷单+微信
	Chinese–English mixture	刷单+V
Syntactic variants	Inversion of word order	刷单加信微

categories: correction models (Liu SL et al., 2021; Sheng et al., 2023) and machine translation models (Stahlberg, 2020; Dabre et al., 2021). The correction models generally consist of the detection task and the correction task (Zhang SH et al., 2020). These models need to identify errors from the source text and then obtain the normal text by reasonable replacement. Therefore, this approach is more suitable for search engines and speech recognition tasks that contain a few spelling errors. In contrast to spelling error correction, variant characters are similar in sound and form to the corresponding normal characters, but the meanings are very different. The intricate and myriad variation features significantly escalate the challenges associated with detection tasks. Using merely correction models does not allow for a profound understanding of the variant characters within diverse scripts. Machine translation is an important and effective method for dealing with the recognition of Chinese variant characters (Bryant et al., 2023). However, the input and output of traditional machine translation models are mostly in two different languages, while the input and output of the Chinese variant-character-recognition task are both in Chinese, with extremely similar or even identical text lengths and characters. In addition, ordinary characters play a crucial role in the meaning of the entire variant text, which requires the model to have better mining and capturing capabilities of key information in the text and to fully understand the semantics of the entire text.

The burgeoning interest in natural language processing (NLP) tasks has prompted a growing number of researchers to shift their focus toward the Chinese language. Since the presentation of Transformer, bidirectional encoder representations from Transformers (BERT), and other models, many NLP models have emerged and have been applied to Chinese domain research (Liu WJ et al., 2020; Maruf et al., 2022). Some scholars have trained large-scale pretraining models in Chinese corpora, which have performed well in the field of Chinese NLP (Diao et al., 2020; Jia C et al., 2020). Some scholars have improved the masked language model (MLM) subtask for the BERT model's pretraining task to make it more applicable to the Chinese language. Sun Y et al. (2019) proposed the novel linguistic representation model Enhanced Representation through kNowledge IntEgration (ERNIE) for knowledge enhancement using Chinese phrase- and entity-level masking strategies to improve the generalizability of the model. Cui et al. (2021) proposed MLM as a correction BERT (MacBERT), a model based on Word2Vec's Chinese similar word replacement instead of MASK, the random masking strategy. However, this Chinese NLP model lacks the analysis of the phonological and morphological features, resulting in insufficient utilization of Chinese character features. A large amount of research in the following years has focused on the features of Chinese characters to explore relevant models. Sun ZJ et al. (2021) represented characters with Chinese Pinyin and glyph embedding and achieved state-of-the-art results in a wide range of Chinese NLP tasks. However, they are not suitable for application to machine translation tasks. These methods prove that the phonological and morphological features of Chinese characters contribute to the field of Chinese NLP when performing Chinese language processing.

The majority of NLP models were initially developed for English text, thus rendering them more attuned to Latin languages, including English and French. In contrast to Latin, Chinese characters have evolved into the current simplified Chinese

throughout history with rich pictographs, structures, and radical features. In addition, Chinese characters have Latin features, i.e., Pinyin, which is the official Romanization system for Standard Mandarin Chinese (https://en.wikipedia.org/wiki/Pinyin). Based on this point, attention needs to be paid to the relationships among the phonological features, morphological features, and the model output. For Chinese variant characters, most of them are transformed based on their phonological and morphological features and contain some complex symbolic variant modes, as shown in Table 1. Therefore, Chinese character phonological and morphological features have a large impact on the Chinese variant-character-recognition task.

Our paper provides a solution to addressing these issues, namely, the shared-weight multimodal translation model (SMTM) based on phonology and morphology. To our knowledge, this model represents the first attempt to integrate phonological and morphological features into the embedding layer of a Chinese variant-character-recognition model using BERT word embeddings, aiming to deeply explore the characteristics of variant characters and enhance the model's learning capabilities. We use the shared-weight embedding mechanism for the Chinese variant character task, initializing the decoder and generator weights using multimodal weights from the source text, thereby improving the model's adaptability to similar text. The main contributions of our paper are as follows:

1. To address the challenges of semantic ambiguity and confusion inherent in Chinese variant characters, we integrate character-level phonological and morphological features using BERT embedding, thereby enhancing the efficacy of unimodal solutions.

2. We propose a shared-weight embedding mechanism to initialize decoder and generator weights based on the multimodal weights of the source text, strengthening the connection between source and target characters, thereby facilitating target sentence generation.

3. We perform simulations using the variant dataset, yielding extensive results that demonstrate the attainment of several state-of-the-art performances by our method. Furthermore, we conduct ablation simulations to offer a reasoned interpretation of our model's performance.

2 Related works

This section provides an elaborate discussion on the pertinent literature concerning the recognition of Chinese variant characters. We review the development and applications of Chinese NLP tasks in Section 2.1. In Section 2.2, we provide an overview of the Chinese variant-character-recognition task and existing solutions. Section 2.3 summarizes the latest research in Chinese variant-character-recognition task based on character phonology and morphology.

2.1 Chinese NLP tasks

In recent years, the Chinese language has emerged as a prominent subject of research, propelled by the evolution of social platforms and online communities. Most NLP models are designed for Latin languages, such as English, French, and Spanish. In contrast to Latin, Chinese has both pictographs and Latin alphabets, e.g., Chinese Pinyin (https://en.wikipedia.org/wiki/Pinyin), radicals (https://en.wikipedia.org/wiki/Radical_(Chinese_characters)), and character structures (https://en.wikipedia.org/wiki/Chinese_characters). Initially, researchers focused on Chinese NLP tasks and improved the convolutional neural network (CNN), recurrent neural network (RNN), and sequence-to-sequence (Seq2Seq) models to make them more suitable for the Chinese language domain (Jia YZ and Xu, 2018; Wu et al., 2019; Zhang YS et al., 2019; Liang et al., 2020). Liu J et al. (2019) combined the features of CNNs, attention mechanisms, and RNNs to propose a bidirectional gated recurrent unit CNN (BiGRU-CNN) network, and achieved the best performance in Chinese problem classification tasks. Yao and Huang (2016) used a bidirectional long short-term memory (BiLSTM) network to create hierarchical feature representations of contextual information from two directions, and achieved the best performance for word segmentation in traditional and simplified Chinese datasets. Ma et al. (2018) supervised the internal representation of Seq2Seq and the representation of the autoencoder by minimizing the distance between two representations, and the results showed that the model outperformed state-of-the-art baseline models.

With the rise of Transformer and BERT, many scholars have been inspired to improve and apply

them to Chinese NLP tasks (Zhou SY et al., 2018; Reimers and Gurevych, 2019; Li XN et al., 2020; Li B et al., 2023). Yan et al. (2019) proposed an adaptive Transformer encoder to model character-level features, which showed the most advanced performance on English and Chinese named-entity recognition (NER) datasets. Liu JG et al. (2020) combined the advantages of BERT and LSTM networks and proposed the BERT ensemble LSTM-BERT (BERT-LB) model, which successively used the BERT model and LSTM model to compute news text features and perform classification based on them; this model outperformed other baseline methods. Chang et al. (2021) designed the BERT-BiLSTM iterated dilated convolutional neural networks (IDCNN)-conditional random fields (CRF) model to solve the problems of multiple meanings of words, insufficient use of contextual information, and easy neglecting of local features, proving the effectiveness of the model in the field of Chinese NER. In contrast to traditional Chinese NLP tasks, most Chinese variant characters are substituted for normal characters maliciously by Chinese phonological and morphological features, resulting in a substantial difference in semantics. This does not affect native speakers' understanding of the actual meaning, but it poses a great challenge to the detection algorithm. Although the above Chinese NLP models have demonstrated certain effectiveness in the Chinese language domain, they are not suitable for the Chinese variant-character-recognition task currently due to the lack of attention to the Chinese character features.

2.2 Chinese correction and translation models

The Chinese variant-character-recognition task (Choi et al., 2018), one of the spelling error correction tasks, is a character-level text replacement technology that uses the computer to recognize Chinese variant text automatically according to a certain application. There are two main representative models: correction models (Wang DM et al., 2018; Hong et al., 2019; Cheng et al., 2020) and machine translation models (Cho et al., 2014; Meng FD et al., 2016; Choi et al., 2018). Many scholars have conducted in-depth research on these two types of models.

The correction models identify variant words from the text and perform the reasonable replacement for variant text by normal text (Chollampatt et al., 2016; Ji et al., 2017; Liu SL et al., 2021; Xu et al., 2021). Therefore, this approach generally consists of two subtasks: detection and conversion. Bao et al. (2020) proposed a chunk-based global optimal decoding method to correct the spelling errors of single- and multi-character words uniformly. Wang DM et al. (2019) provided a confusion set to guide character generation, using the Seq2Seq model for jointly learning to copy the correct characters through a pointer network or to generate a character from the confusion set. Zhao H et al. (2017) proposed a graph model to generate a directed acyclic graph (DAG) for each sentence, and executed a single-source shortest-path algorithm on the graph to simultaneously detect and correct general spelling errors. Zhang SH et al. (2020) designed an error detection network and a BERT-based error correction network based on the two subtasks of the error correction model, namely, detection and correction. The correction model is the most effective and direct variant-character-recognition model, and has achieved impressive results in spelling error correction tasks. However, when the number of variant words in the text is large, the detection task becomes a great challenge.

Traditional machine translation models take one language as input. After embedding, encoding, and decoding, they finally output another language (Zhou J et al., 2020; Soydaner, 2022; Li B et al., 2024). Initially, machine translation models were mainly based on RNN (Zhang B et al., 2020), LSTM (Li WS et al., 2022), and Seq2Seq (Otter et al., 2021) models. Meng FD and Zhang (2019) proposed a deep transition RNN-based machine translation model architecture, and enhanced information mining between hidden layers by multiple nonlinear transformations. Wang YG et al. (2017) demonstrated the Sogou machine translation model which follows the encoder–decoder network structure, implemented a deep multilayer LSTM network for the encoder and decoder, and finally achieved satisfactory performance on a public Chinese–English news translation dataset. The attention-based and pretrained models have attracted much interest from researchers recently. Chen et al. (2018) used syntactic distance constraints to extend local attention to learn more efficient context vectors to predict translations, and thereby achieved considerable improvements in Chinese–English and English–German

translation tasks. Liu Y and Lapata (2019) introduced an attention mechanism based on Transformer to represent cross-document relationships. The use of fine-tuning in neural machine translation (NMT) does not fully use prior language knowledge; to address this problem, Weng et al. (2020) designed an acquiring knowledge from pretrained model for knowledge acquisition from pretrained models to NMT. The results of the above methods have made great contributions to Chinese machine translation tasks. In contrast to conventional machine translation tasks, Chinese variant word tasks involve input and output within the same language, exhibiting a degree of similarity. Consequently, it is unsurprising that traditional methods exhibit relatively weak performance on these tasks.

2.3 Chinese NLP models based on Pinyin and fonts

Early models based on Chinese Pinyin and glyph features were mainly used in text classification (Zhuang et al., 2017; Xie et al., 2020), natural language inference (NLI) (Hu et al., 2020; Shao et al., 2024), and NER (Li JT and Meng, 2021; Zhao S et al., 2023). Zhang Y et al. (2019) proposed a new method, ssp2vec, to predict the contextual words based on the feature substrings of the target words for learning Chinese word embedding, and found that both morphological information (strokes and structure) and phonological information (Pinyin) are crucial for learning the meaning of Chinese words. Inspired by computer vision, Dai and Cai (2017) used grayscale images of Chinese characters as the input of CNN embedders, and mapped them to a k-dimensional vector space through a fully connected layer to represent the Chinese character glyphs. Meng YX et al. (2019) designed a Tianzige-CNN structure suitable for Chinese character image processing, and used three types of font images, namely, bronzeware script, seal script, and traditional Chinese characters, to extract the glyph features of Chinese characters. For the pictograph features and radical structure of Chinese characters, Tao et al. (2019) demonstrated an attention-based four-granularity model to fully use the four features of the Chinese language, namely, characters, words, character-level radicals, and word-level radicals, and designed an attention mechanism to enhance the attention of radicals by using BiLSTM to sense the contextual context. Considering the methodologies mentioned earlier, the integration of Chinese phonological and morphological features proves advantageous to the model's performance in NLP tasks, thereby affirming the imperative of using Pinyin and glyphs as features.

3 SMTM approach

We attribute the variant-character-recognition task to character-level machine translation. Dataset D contains pairs of variant text X^{V} and normal text X^{N}. The goal is to build a character-level Chinese NMT model that generates normal text $n \in X^{\mathrm{N}}$ based on variant $v \in X^{\mathrm{V}}$.

In this section, we demonstrate the details of the Chinese variant-character-translation model based on phonology and morphology, namely, the SMTM. The model contains an embedding layer, an encoding layer, and a decoding layer. First, we follow the fusion embedding introduced by Sun ZJ et al. (2021), and propose a shared-weight multimodal embedding layer to make full use of the phonological and morphological features of Chinese characters (Section 3.1). Second, our model uses the encoder-decoder architecture (Chen et al., 2018), in which the encoder encodes the input text into hidden representations, and the decoder generates the target text based on these representations (Sections 3.2 and 3.3). Fig. 1 shows the model's architecture. The main notations used in this paper are listed in Table 2.

3.1 Shared-weight multimodal embedding layer

Our model adds Pinyin embedding and font embedding on top of BERT embedding to extract multimodal features from variant text in the dataset. We use the shared-weight embedding mechanism to handle similar text.

3.1.1 BERT embedding

We follow the BERT embedding layer proposed by Devlin et al. (2019), which contains token embedding, position embedding, and segment embedding. The tokenizer converts the input text into vector form, and adds [CLS] and [SEP] before and after the sentence, respectively, and then feeds them into token embedding. Segment embedding distinguishes

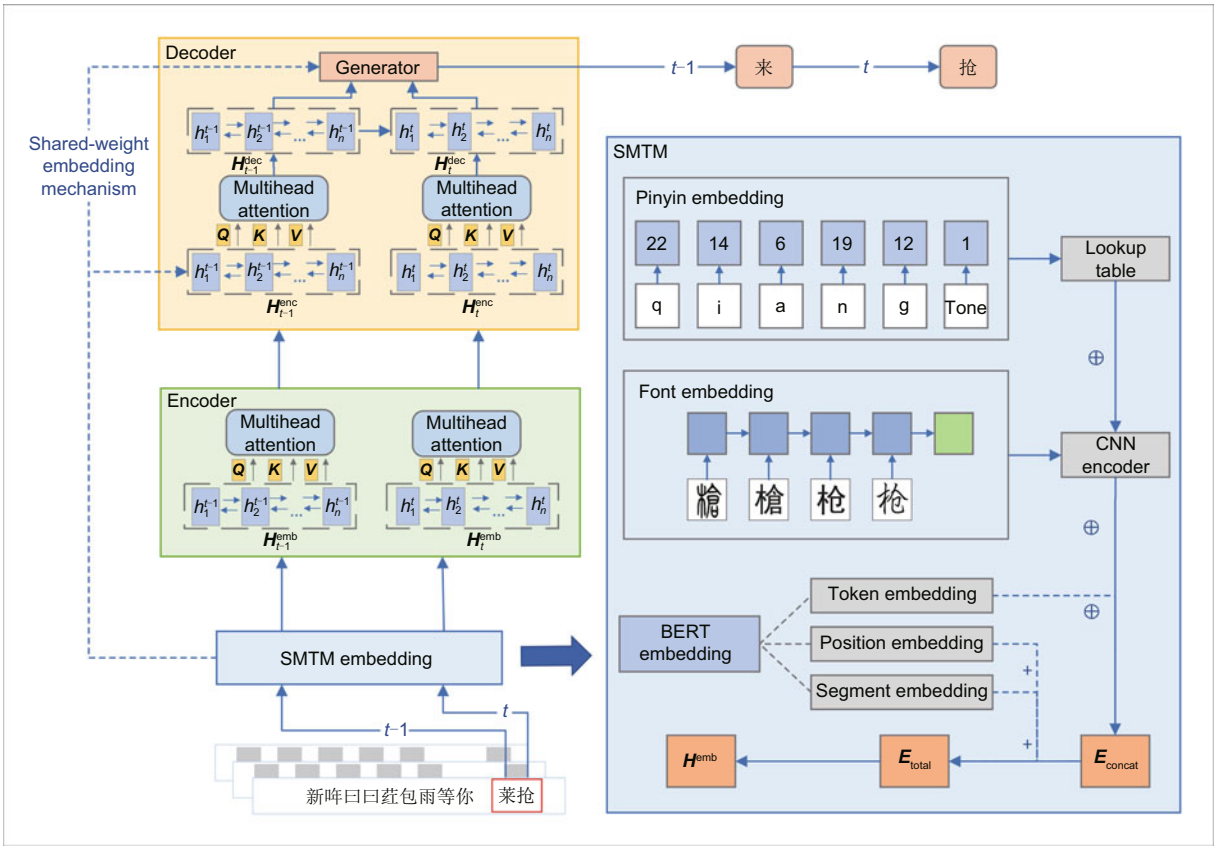

Fig. 1 Illustration of the SMTM architecture, where ⊕ denotes vector concatenation

Table 2 Main notations used in this paper

Notation	Description
X^{V}, X^{N}	Variant and normal sets, respectively
v, n	Variant and normal sentences in the corresponding set, respectively
$\boldsymbol{E}_{\mathrm{token}}, \boldsymbol{E}_{\mathrm{pos}}, \boldsymbol{E}_{\mathrm{seg}}$	Token embedding, position embedding, and segment embedding in the BERT model, respectively
$\boldsymbol{c}_i^{\mathrm{token}}$, $\boldsymbol{c}_i^{\mathrm{pos}}$, $\boldsymbol{c}_i^{\mathrm{seg}}$	The i^{th} character of the variant text in $\boldsymbol{E}_{\mathrm{token}}$, $\boldsymbol{E}_{\mathrm{pos}}$, and $\boldsymbol{E}_{\mathrm{seg}}$, respectively
$\boldsymbol{E}_{\mathrm{pinyin}}$, $\boldsymbol{E}_{\mathrm{font}}$	Pinyin and font embeddings, respectively
$\boldsymbol{c}_i^{\mathrm{pinyin}}$, $\boldsymbol{c}_i^{\mathrm{font}}$	The i^{th} character of the variant text in the Pinyin and font embeddings, respectively
l, l_{voc}	Lengths of the variant text and vocabulary, respectively
d_{emb}, d_{hid}	Dimensions of the embedding and hidden layers, respectively
$\mathrm{PE}_{(i,2k)}$	The $(2k)^{\mathrm{th}}$ dimension of the i^{th} character in the position encoding
$\boldsymbol{W}_{\mathrm{gen}}$, $\boldsymbol{W}_{\mathrm{dec}}$, $\boldsymbol{W}_{\mathrm{enc}}$	Initial weights of the generator, decoder, and encoder, respectively
$\boldsymbol{H}_{\mathrm{v}}$	Output state of the variant by normalization and dropout layer in the encoder
$\boldsymbol{Q}$, $\boldsymbol{K}$, $\boldsymbol{V}$	Query matrix, key matrix, and value parameter matrix, respectively
$\boldsymbol{H}_t^{\mathrm{enc}}$, $\boldsymbol{H}_t^{\mathrm{dec}}$	Output states of the encoder and decoder in the t^{th} step, respectively

two independent sentences by "0" and "1." Position embedding helps the model focus on the same word in different positions in a sentence, and enhances the sensitivity of the position.

We use token embedding, position embedding, and segment embedding, denoted by $\boldsymbol{E}_{\mathrm{token}} = (\boldsymbol{c}_1^{\mathrm{token}}, \boldsymbol{c}_2^{\mathrm{token}}, \ldots, \boldsymbol{c}_l^{\mathrm{token}}) \in \mathbb{R}^{l\times d_{\mathrm{emb}}}$, $\boldsymbol{E}_{\mathrm{pos}} = (\boldsymbol{c}_1^{\mathrm{pos}}, \boldsymbol{c}_2^{\mathrm{pos}}, \ldots, \boldsymbol{c}_l^{\mathrm{pos}}) \in \mathbb{R}^{l\times d_{\mathrm{emb}}}$, and $\boldsymbol{E}_{\mathrm{seg}} = (\boldsymbol{c}_1^{\mathrm{seg}}, \boldsymbol{c}_2^{\mathrm{seg}}, \ldots, \boldsymbol{c}_l^{\mathrm{seg}}) \in \mathbb{R}^{l\times d_{\mathrm{emb}}}$, respectively, where $\boldsymbol{c}_i^{\mathrm{token}}$, $\boldsymbol{c}_i^{\mathrm{pos}}$, and $\boldsymbol{c}_i^{\mathrm{seg}}$ denote the token vector, position vector, and segment vector corresponding to the i^{th} character of the variant text, respectively. d_{emb} is the embedding layer dimension, and l represents the variant text length.

$\boldsymbol{E}_{\text{pos}}$ adopts absolute position embedding and uses sinusoidal position encoding (Vaswani et al., 2017), denoted by PE, as shown in Eqs. (1) and (2); $\text{PE}_{(i,2k)}$ and $\text{PE}_{(i,2k+1)}$ are the $(2k)^{\text{th}}$ and $(2k+1)^{\text{th}}$ dimensions of the i^{th} character in the position encoding, respectively; k is the vector dimension.

$$\text{PE}_{(i,2k)} = \sin(i/10\,000^{2k/d_{\text{emb}}}), \tag{1}$$

$$\text{PE}_{(i,2k+1)} = \cos(i/10\,000^{2k/d_{\text{emb}}}). \tag{2}$$

3.1.2 Pinyin embedding

Chinese has a unique language system, Pinyin. Pinyin is a modern standard Chinese phonetic scheme using the Latin alphabet, which is the international standard specification for modern standard Chinese Romanization. Pinyin consists of 23 initials, 24 finals, and 5 tones. The longest length in Chinese characters is six (e.g., the Chinese character, "庄," is represented in Pinyin as zhuāng). Unlike Chinese Pinyin correction models, the Pinyin embeddings for variant characters in our approach are consistent with those of the standard characters. These embeddings are combined with contextual encoding to capture the semantic information of the entire sentence. Therefore, a vector of a fixed length of eight is used to represent the corresponding Pinyin of Chinese characters. The model uses the PyPinyin package (https://pypi.org/project/pypinyin/) to encode the Pinyin letters and tones. It appends the tones to the end of the vector, i.e., $\boldsymbol{c}_i^{\text{pinyin}} = [\text{Initials, Finals, Tone}] \in \mathbb{R}^{1\times 8}$. Here, $\boldsymbol{c}_i^{\text{pinyin}}$ denotes the Pinyin vector corresponding to the i^{th} character of the variant text. If the length is <8, zero is the placeholder for the vector. $\boldsymbol{E}_{\text{pinyin}} = (\boldsymbol{c}_1^{\text{pinyin}}, \boldsymbol{c}_2^{\text{pinyin}}, \ldots, \boldsymbol{c}_l^{\text{pinyin}}) \in \mathbb{R}^{l\times d_{\text{emb}}}$ is the Pinyin embedding.

3.1.3 Font embedding

Chinese characters are one of the oldest scripts in the world, with a history of >6000 years, and have gradually changed from complex to simple. We add the traditional font images based on three other fonts, namely, Fangsong, clerical script, and Xingkai, denoted by $\text{img}_i^{\text{font}}$. A random sample of 2000 variant characters is selected, and their stroke counts are analyzed. The relationship between stroke count and Chinese variant character count is illustrated in Fig. 2. As observed, the majority of Chinese variant characters have stroke counts ranging between 10 and 20. According to Li WG et al. (2024), the threshold for effective Chinese character image processing varies with pixel size and stroke count. For characters with stroke counts between 12 and 26, a 24-pixel size is recommended. Consequently, we set the image size to 24 × 24 pixels. We apply a CNN to encode the image of each font as the font embedding $\boldsymbol{E}_{\text{font}} = (\boldsymbol{c}_1^{\text{font}}, \boldsymbol{c}_2^{\text{font}}, \ldots, \boldsymbol{c}_l^{\text{font}}) \in \mathbb{R}^{l\times d_{\text{emb}}}$. $\boldsymbol{c}_i^{\text{font}}$ denotes the font vector corresponding to the i^{th} character of the variant text, as shown in Eq. (3):

$$\boldsymbol{c}_i^{\text{font}} = \text{CNN}(\text{img}_i^{\text{font}}). \tag{3}$$

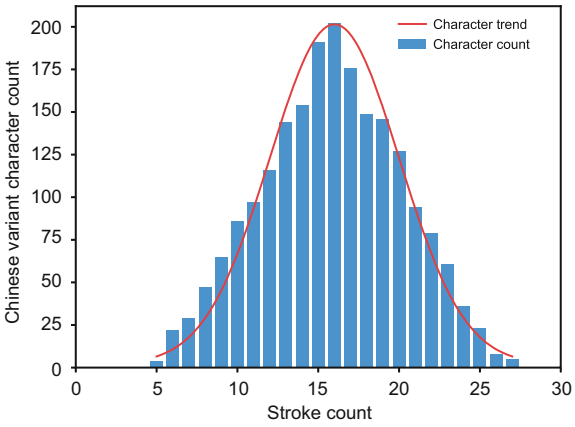

Fig. 2 Distribution of the stroke count and the Chinese variant character count

3.1.4 Emoji enhancement

In the current common word embedding process, emojis and special symbols are represented by [UNK], resulting in insufficient semantic representability of the model. To address this issue, we use the emojiswitch package (https://pypi.org/project/emojiswitch/) for emoji recognition to handle out-of-vocabulary (OOV) text.

3.1.5 Shared-weight embedding mechanism

We use the PyTorch package for model development, which generates a random set of weights for the source sentence, target sentence, and generator by default. The principle is that the decoder generates text based on the embedding result of the target sentence and the predicted text. However, with the increase of the number of training steps, the target sentence weights diverge from the source

sentence weights gradually, making it difficult to accurately mine the semantic information of similar text. Therefore, we propose the shared-weight embedding mechanism to initialize the target sentence weights and generator weights in the decoder, as shown in Eq. (4). This mechanism ensures a one-to-one correspondence between text and embedding in the encoder, decoder, and generator, which enhances the text generation capability of the model.

$$\boldsymbol{W}_{\text{gen}} = \boldsymbol{W}_{\text{dec}} = \boldsymbol{W}_{\text{enc}} \in \mathbb{R}^{l_{\text{voc}} \times d_{\text{emb}}}, \boldsymbol{W}_{\text{enc}} \sim N(0,1), \tag{4}$$

where $\boldsymbol{W}_{\text{gen}}$, $\boldsymbol{W}_{\text{dec}}$, and $\boldsymbol{W}_{\text{enc}}$ denote the initial weights of the generator, decoder, and encoder, respectively. l_{voc} represents the length of the vocabulary. The initial weights obey a normal distribution $N(0,1)$ with a mean of 0 and a variance of 1. The shared-weight embedding mechanism can improve the model's semantic mining capability and handle a high semantic similarity between the input and output. The detailed embedding module is shown in Algorithm 1.

Algorithm 1 Shared-weight multimodal embedding module

Require: source sentence of Chinese variant characters, **sentence**=$(\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_l)$
Ensure: the vector after the extraction of phonological and morphological features, $\boldsymbol{c}_i^{\text{total}}$

for $i = 0$ **to** len(**sentence**) **do**
  $\boldsymbol{c}_i^{\text{pinyin}} = \text{pinyinProcess}(\boldsymbol{c}_i)$;
  while $\text{len}(\boldsymbol{c}_i^{\text{pinyin}}) \leq 8$ **do**
    $\boldsymbol{c}_i^{\text{pinyin}}$.append(0);
  end while
  for $j = 0$ **to** 4 **do**
    $\boldsymbol{c}_j^i = \text{CNN}(\boldsymbol{c}_i)$;
    $\boldsymbol{c}_i^{\text{font}}$.append$(\boldsymbol{c}_j^i)$;
  end for
  $(\boldsymbol{c}_i^{\text{token}}, \boldsymbol{c}_i^{\text{pos}}, \boldsymbol{c}_i^{\text{seg}}) \Leftarrow \text{BERTEmbProcess}(\boldsymbol{c}_i)$;
  $\boldsymbol{c}_i^{\text{concat}} = \text{Concat}[\boldsymbol{c}_i^{\text{pinyin}}, \boldsymbol{c}_i^{\text{font}}, \boldsymbol{c}_i^{\text{token}}]$; /* concatenated character representation */
  $\boldsymbol{c}_i^{\text{total}} = \boldsymbol{c}_i^{\text{concat}} + \boldsymbol{c}_i^{\text{pos}} + \boldsymbol{c}_i^{\text{seg}}$;
end for
return $\boldsymbol{c}_i^{\text{total}}$

3.1.6 Noise analysis

Regarding the handling of noise in multiple embedding representations (Jin et al., 2022), we optimize the learning process of various Chinese embedding representations through a shared-weight embedding mechanism. Weight sharing is implemented across four different embedding layers, which effectively reduces the total number of model parameters by eliminating the need to train and maintain separate sets of weights for each layer. Meanwhile, the shared-weight embedding mechanism ensures consistency and coherence in the processing of embedding representations across layers, thereby facilitating effective information transfer and integration between different layers. Moreover, by using identical weights to learn and transform input data in each encoding layer, key information is extracted and represented in a coordinated and consistent manner. This coordination helps minimize noise accumulation due to interlayer differences or inconsistencies, enhancing the overall stability and accuracy of the embedding representation. In summary, by enabling different embedding representation layers to share a common set of weights, we not only significantly reduce parameter redundancy but also effectively decrease the noise introduced during the fusion and integration of diverse embeddings. This approach improves the generalization ability of the model and enhances its robustness and accuracy in handling complex Chinese linguistic phenomena.

3.2 Encoder

3.2.1 Encoder input

The output of the embedding layer is used as the input for the encoder. We concatenate the Pinyin and font embeddings with the token embedding and send the result to the fully connected (FC) layer. After summing with position and token embeddings, the result is fed to the normalization and dropout layers to obtain the output of the embedding layer. The specific process is shown in Eqs. (5)–(7):

$$\boldsymbol{E}_{\text{concat}} = \text{FC}(\boldsymbol{E}_{\text{pinyin}} \oplus \boldsymbol{E}_{\text{font}} \oplus \boldsymbol{E}_{\text{token}}), \tag{5}$$

$$\boldsymbol{E}_{\text{total}} = \boldsymbol{E}_{\text{concat}} + \boldsymbol{E}_{\text{pos}} + \boldsymbol{E}_{\text{seg}}, \tag{6}$$

$$\boldsymbol{H}_{\text{v}} = \text{Dropout}(\text{Norm}(\boldsymbol{E}_{\text{total}})), \tag{7}$$

where $\oplus$ denotes vector concatenation, $\boldsymbol{E}_{\text{concat}} \in \mathbb{R}^{l \times d_{\text{emb}}}$ represents the result of the concatenation of Pinyin, font, and token embeddings, $\boldsymbol{E}_{\text{total}} \in \mathbb{R}^{l \times d_{\text{emb}}}$ is the result after the summation of $\boldsymbol{E}_{\text{concat}}$, position embedding, and segment embedding, and $\boldsymbol{H}_{\text{v}} \in \mathbb{R}^{l \times d_{\text{hid}}}$ represents the output of the normalization and dropout layers. The dimension of the hidden layer is denoted by d_{hid}, where $d_{\text{hid}} = d_{\text{emb}}$.

3.2.2 Attention mechanism of the encoder

The complexity and diversity of Chinese variants pose a certain challenge to mining the core semantic information. The attention mechanism can capture the key parts of the vector well. Our model uses a multihead attention mechanism to focus on different positions by giving attention to multiple subrepresentation spaces. The query matrix $\boldsymbol{Q} \in \mathbb{R}^{l\times d_{\text{hid}}}$, key matrix $\boldsymbol{K} \in \mathbb{R}^{l\times d_{\text{hid}}}$, and value matrix $\boldsymbol{V} \in \mathbb{R}^{l\times d_{\text{hid}}}$ are calculated by the query, key, and value parameter matrices of the i^{th} head, $\boldsymbol{W}_i^q \in \mathbb{R}^{d_{\text{hid}}\times d_q}$, $\boldsymbol{W}_i^k \in \mathbb{R}^{d_{\text{hid}}\times d_k}$, and $\boldsymbol{W}_i^v \in \mathbb{R}^{d_{\text{hid}}\times d_v}$, respectively. The MultiHead$(\cdot)$ is obtained based on all the head$_i$ and output weights $\boldsymbol{W}^{\text{O}} \in \mathbb{R}^{d_v\times d_{\text{hid}}}$, as shown in Eqs. (8)–(10), where $i \in \{1,2,...,\mathcal{N}\}$ and $\mathcal{N}$ represents the number of heads. We set the number of attention heads as $h=8$, and $d_q=d_k=d_v=d_{\text{hid}}/h$. The encoder output is obtained after the residual network, normalization layer, and feedforward (FF) network.

$$\boldsymbol{Q}_i=\boldsymbol{H}_{\text{v}}\boldsymbol{W}_i^q,\boldsymbol{K}_i=\boldsymbol{H}_{\text{v}}\boldsymbol{W}_i^k,\boldsymbol{V}_i=\boldsymbol{H}_{\text{v}}\boldsymbol{W}_i^v, \tag{8}$$

$$\text{head}_i(\boldsymbol{Q}_i,\boldsymbol{K}_i,\boldsymbol{V}_i)=\text{softmax}\left(\frac{(\boldsymbol{Q}_i\boldsymbol{K}_i)^{\text{T}}}{\sqrt{d_k}}\right)\boldsymbol{V}_i, \tag{9}$$

$$\text{MutiHead}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V})=(\text{head}_1\oplus\text{head}_2\oplus\cdots\oplus\text{head}_{\mathcal{N}})\boldsymbol{W}^{\text{O}}. \tag{10}$$

3.3 Decoder

3.3.1 Decoder input

The input of the decoder contains two parts: the output of the decoder at the previous time step and the output of the last encoder at the current time step, both of which are obtained from the output of the multihead attention mechanism through the residual connection, regularization layer, and FF layer, as shown in Eqs. (11) and (12). Additionally, the encoder output is a set of sequence vectors, which are used as the input of the decoder's key matrix and value parameter matrix.

$$\boldsymbol{H}_t^{\text{enc}}=\text{FF}(\text{Add\&Norm}(\text{MultiHead}_t^{\text{enc}}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}))), \tag{11}$$

$$\boldsymbol{H}_{t-1}^{\text{dec}}=\text{FF}(\text{Add\&Norm}(\text{MultiHead}_{t-1}^{\text{dec}}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}))), \tag{12}$$

where $\boldsymbol{H}_t^{\text{enc}}$ denotes the output of the encoder at time step t and $\boldsymbol{H}_{t-1}^{\text{dec}}$ is the output of the decoder at the previous time step $t-1$.

3.3.2 Attention mechanism of the decoder

In contrast to the encoder, in the attention mechanism of the decoder, the decoder adds a masked multihead attention mechanism, which enables the model to focus only on words earlier than the current position in the output sequence by positional encoding. The input of the query matrix $\boldsymbol{Q}_t^{\text{dec}}$ is obtained based on the output $\boldsymbol{H}_{t-1}^{\text{dec}}$ of the decoder at time step $t-1$. The input of the key matrix $\boldsymbol{K}_t^{\text{dec}}$ and value matrix $\boldsymbol{V}_t^{\text{dec}}$ is calculated from the output of the encoder $\boldsymbol{H}_t^{\text{enc}}$, as shown in Eqs. (13)–(15). Through the attention mechanism, it is beneficial to focus on the relationship between the predicted normal character and the current variant character to mine deeper semantic information and enhance the learnability of the model on the Chinese variant-character-recognition task.

$$\boldsymbol{K}_t^{\text{dec}}=\boldsymbol{H}_t^{\text{enc}}\boldsymbol{W}_t^k, \tag{13}$$

$$\boldsymbol{V}_t^{\text{dec}}=\boldsymbol{H}_t^{\text{enc}}\boldsymbol{W}_t^v, \tag{14}$$

$$\boldsymbol{Q}_t^{\text{dec}}=\boldsymbol{H}_{t-1}^{\text{dec}}\boldsymbol{W}_t^q. \tag{15}$$

3.3.3 Generator layer

The generator calculates the vocabulary probability distributions based on the log softmax layer and linear layer according to attention scores to obtain the corresponding predicted word, as shown in Eq. (16):

$$\boldsymbol{P}_{\text{vocab}}=\text{LogSoftmax}(\boldsymbol{W}_1\boldsymbol{\alpha}_t+\boldsymbol{b}_1), \tag{16}$$

where $\boldsymbol{P}_{\text{vocab}}$ is the vocabulary probability distribution of the task, $\boldsymbol{\alpha}_t$ denotes the attention score at step t, and $\boldsymbol{W}_1$ and $\boldsymbol{b}_1$ are learnable parameters.

4 Simulations

4.1 Datasets and evaluation metrics

4.1.1 Datasets

We evaluate our model on the CCF BDCI dataset, which is from the BDCI competition, and the data consist of various spam text contents reported by users. The variant mode, including morphological similarities, homophones, traditional characters, and various interfering characters, is inserted into the text for spacing. The datasets contain 60 915 pieces of text; 11 358 pieces of data are used

for training and 49 557 pieces of data are used for test. Each piece of data consists of source and target sentences, corresponding to variant and normal text, respectively; their average lengths are 54 and 49, respectively.

4.1.2 Evaluation metrics

This paper uses the bilingual evaluation understudy (BLEU) metric (Papineni et al., 2002) to measure the fluency of translations and the F1 metric to demonstrate the correction of variant characters. BLEU is a fast, efficient, and language-independent automatic machine translation evaluation metric. It measures the fluency and similarity between the candidates and the references to obtain the corresponding score. Therefore, we calculate BLEU_1 and BLEU_2, i.e., single-word overlap and binary overlap in the text, and use two sets of weights $[0.4, 0.6, 0, 0]$ and $[0.25, 0.25, 0.25, 0.25]$ to obtain BLEU_{1-2} and $\mathrm{BLEU}_{\mathrm{avg}}$, respectively. We calculate the score after averaging with F1, as shown in Eqs. (17)–(20):

$$\mathrm{BLEU} = \mathrm{BP} \cdot \mathrm{e}^{\sum_{n=1}^{N}(w_n \log p_n)} \times 100\%, \tag{17}$$

$$\mathrm{BP} = \begin{cases} \mathrm{e}^{1-\frac{r}{c}}, & c \le r, \\ 1, & c > r, \end{cases} \tag{18}$$

$$\mathrm{F1} = 2 \times \frac{\mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision}+\mathrm{Recall}} \times 100\%, \tag{19}$$

$$\mathrm{Score} = \frac{(\mathrm{BLEU}_{1-2} + \mathrm{BLEU}_{\mathrm{avg}})/2 + \mathrm{F1}}{2} \times 100\%, \tag{20}$$

where r and c denote the numbers of references and candidate translations respectively, BP and p_n represent the penalty factor and the value of n-gram, N denotes the number of grams ($n \in \{1, 2, \cdots, N\}$), and w_n is the weight of n-gram, which adds up to 1.

4.2 Implementations and baseline methods

4.2.1 Implementations

We use the PyTorch and Transformer package to implement the SMTM and extend the size of the BERT-based Chinese dictionary from 21 128 to 23 236 to better handle OOV text. The model adopts the Transformer encoder–decoder architecture, which consists of 14 layers with eight attention heads. The number of hidden units is 768, and the hidden size of all FF layers is 1024. The embedding layer uses a shared-weight multimodal embedding layer, where the position embedding layer is initialized with the position embedding of BERT and uses the source sentence weights to initialize the target sentence weights and generator weights of the decoder during training. We train our model with the Adam optimizer for 31 800 steps, the batch size is set to 10, the dropout is set to 0.1, and the initial learning rate is set to 0.4 for the best results. We also apply the learning rate warmup over the first 2385 steps to improve the stability of the model.

4.2.2 Baseline methods

We select several baseline models with representatives for comparison, as follows:

1. Error correction model. The Chinese variant-character-recognition task is similar to the Chinese spelling correction (CSC) task. So, we try to conduct comparison using the error correction model to compare the performance of the two different models in the current task. We choose the confusion set-guided pointer network (CPN) (Wang DM et al., 2019), the most advanced model in the field of CSC, which can recognize variant characters using the Chinese character confusion set for the comparison.

2. Seq2Seq. Seq2Seq is a common and effective method in the field of machine translation, which includes two parts: encoder and decoder. We choose three different Seq2Seq models to compare the model effect, namely, recurrent Seq2Seq, convolutional Seq2Seq (Gehring et al., 2017), and state-of-the-art pretrained model BERT encoder, represented by BBT (BERT embeddings + BERT encoder + Transformer decoder), to observe the performance of the pretrained model on the dataset. The BERT model was developed by the HuggingFace team (https://huggingface.co/bert-base-chinese/tree/main) and has been pretrained in the Chinese corpus.

3. Embedding layer. The embedding layer is designed to extract textual features. We choose the most common and advanced embedding layers for comparison, namely, random embedding and BERT embedding. The Transformer encoder–decoder architectures, i.e., RTT (random embedding + Transformer encoder + Transformer decoder) and BTT (BERT embedding + Transformer encoder + Transformer decoder) (Liu Y and Lapata, 2019), are used as comparison to further illustrate the advanced nature of the SMTM.

4. Large language model. Large language models have become one of the most prominent research directions and represent the latest advancements in NLP. Due to hardware constraints, the open-source models Qwen2-7B (Yang et al., 2024) and Llama 3.1 8b (Dubey et al., 2024) are selected as the baseline models for our comparison. These models are used to demonstrate the superiority of the proposed SMTM.

4.3 Simulation results and comparisons

We evaluate the model's performance using BLEU_{1-2}, BLEU_{avg}, and F1 metrics. Table 3 illustrates the performance of the proposed method and the baseline models. It can be seen that the SMTM performs superiorly, which is consistent with the results of the methodological study, as detailed in the following observations and analysis:

In contrast to the error correction model, machine translation models demonstrate superior efficacy in recognizing Chinese variant characters. Simulation results demonstrate that our model significantly surpasses the error correction model. Prediction outcomes reveal that when the text contains numerous variant text, the error correction model performs poorly, corroborating previous research findings. Furthermore, the CPN, which primarily uses Chinese character confusion sets for assistance in transformation, is limited in scope and cannot encompass all variants or learn non-Chinese characters. Consequently, the error correction model lacks the capability to recognize variant characters adequately.

For the Seq2Seq approach, Table 3 indicates that the attention-based Seq2Seq model (Transformer and BERT) surpasses convolutional and recursive methods, suggesting that the semantics of the target text is inferred from the contextual core information of the source text. Furthermore, when compared to Transformer, the Seq2Seq model using BERT as the encoder demonstrates a decline in performance. The simulation results reveal that the BERT encoder relies on extensive downstream prediction training, and when juxtaposed with more structured pretraining corpora, there are notable disparities between the BERT Chinese corpus and the tasks addressed in this simulation.

The embedding layer exerts the most pronounced influence on the model, with the SMTM attaining superior performance in both BLEU and F1 metrics. The primary disparities among the simulation outcomes of the models are manifested in the F1 metric, with the SMTM exhibiting an improvement of 0.474 percentage points (PPs) over the optimal baseline. Given that normal characters correspond to multiple variant characters, the correlation of features among characters is crucial for the semantic mining of characters. Among all the commonly used embedding techniques, our model can learn a greater number of Chinese character features, thereby accurately identifying and transforming variant words.

Large language models have demonstrated remarkable capabilities in tasks such as dialogue and writing assistance. However, their performance is somewhat limited in recognizing and correcting Chinese variant characters. As shown in Table 3, the proposed SMTM outperforms the baseline large language models across all metrics. The lower accuracy in recognizing Chinese variants by Llama 3.1 can be attributed to its limited inclusion of Chinese semantic information. Although Qwen2 has

Table 3 Performance comparison on the CCF BDCI dataset

Type	Model	BLEU_{1-2} (%)	BLEU_{avg} (%)	F1 (%)	Score (%)
Error correction model	CPN	79.767	75.849	70.134	73.971
Seq2Seq	Recurrent Seq2Seq	70.690	65.886	65.510	66.899
	Convolutional Seq2Seq	72.269	68.533	66.667	68.534
	BBT	87.333	83.150	67.027	76.134
Embedding layer	RTT	85.206	80.745	76.392	79.684
	BTT	85.121	80.832	79.006	80.991
Large language model	Llama 3.1 8b	72.871	67.609	65.928	68.084
	Qwen2-7B	81.537	77.132	75.264	77.299
Our model	SMTM	**89.550**	**86.017**	**79.480**	**83.632**

The best results are in bold

undergone extensive pretraining on Chinese semantics, its performance on Chinese variant characters is suboptimal. This is because recognizing these variants requires not only an understanding of Chinese semantics but also reasoning about Chinese Pinyin, font, and emoji. Our model, on the other hand, enhances performance by learning Chinese semantics, Pinyin, font, and emoji. The shared-weight embedding mechanism enables a deeper exploration of the differences and correlations between Chinese variant characters and standard text, resulting in superior performance.

5 Ablation studies

5.1 Effect of each SMTM component

In this subsection, we detail ablation simulations conducted under various settings to investigate the influence of each component in SMTM on model effectiveness. Table 4 displays the scores of different components of the ablation simulations on the dataset. The primary objective of this study is to uncover the semantic information within variant text using Pinyin and font embeddings, along with the shared-weight embedding mechanism. Our findings reveal that the SMTM achieves the highest score, demonstrating its superior performance in the task of Chinese variant character transformation. Furthermore, removing any component results in varying degrees of model degradation. We remove Pinyin embedding, font embedding, and shared-weight embedding mechanism, to analyze which factor leads to a decrease in the performance of the model.

1. W/o Pinyin embedding (Here w/o means without). Removing Pinyin embedding has the greatest impact on the model, decreasing the model performance score by 2.658 PPs and the F1 metric by 3.761 PPs. It illustrates that Pinyin embedding has the greatest contribution to the SMTM model.

2. W/o font embedding. Removing the font embedding affects the model's efficiency. We can observe that the F1 metric of the translation increases considerably instead. An analysis of the data associated with the improved F1 metric indicates that in approximately 76% of the cases, non-Chinese characters account for more than 50% of the content. This is because, after removing the font embedding, variant characters with specific Chinese font features cannot be accurately recognized or corrected. As a result, both the numerator and the denominator in the precision calculation of Eq. (19) decrease, with a more significant drop in the denominator, leading to an increase in the precision value. Consequently, a slight increase in the F1 metric is observed in Table 4, as indicated by Eq. (19). Despite the increase in the F1 metric, the score decreases overall. Similarly, this is the reason for the rise in the F1 in other ablation simulations.

3. W/o the shared-weight embedding mechanism for the decoder. It is worth noting that removing the shared-weight embedding mechanism for the decoder results in a larger decrease in BLEU-related metrics, with a maximum decrease of 3.550 PPs. Analysis of the average percentage of identical characters between source and target sentences reveals a higher similarity in the data where BLEU metric decreases, with the proportion increasing by more than one-tenth relative to the alternative variant. This indicates that the performance of the model decreases in the scene of similar text after removing the shared-weight embedding mechanism.

4. W/o the shared-weight embedding mechanism for the generator. Compared with removing the shared-weight embedding mechanism for the decoder, the change in score after removing the shared-weight embedding mechanism for the generator is smaller. We can analyze that the generator structure

Table 4 Results of SMTM model ablation simulations

Model	$BLEU_{1-2}$ (%)	$BLEU_{avg}$ (%)	F1 (%)	Score (%)
SMTM	**89.550**	**86.017**	79.480	**83.632**
W/o Pinyin embedding	88.144 (↓1.406)	84.314 (↓1.703)	75.719 (↓3.761)	80.974 (↓2.658)
W/o font embedding	87.009 (↓2.541)	83.164 (↓2.853)	**81.859** (↑2.379)	83.473 (↓0.159)
W/o sharing for decoder	87.167 (↓2.383)	82.467 (↓3.550)	80.092 (↑0.612)	82.455 (↓1.177)
W/o sharing for generator	87.245 (↓2.305)	82.732 (↓3.285)	79.976 (↑0.496)	82.482 (↓1.150)
W/o emoji enhancement	87.663 (↓1.887)	83.455 (↓2.562)	80.299 (↑0.819)	82.929 (↓0.703)

The best results are in bold. The value in the brackets means the improvement or decrease compared with the SMTM, denoted by ↑ and ↓, respectively

is simpler and the shared-weight embedding mechanism affects only the linear and softmax layers and has less impact on the learning and mining of the semantics during model training.

5. W/o emoji enhancement. We remove the emoji enhancement in the embedding process. It has some effect on the special scene of the emoji. It can be seen that there is a decrease in both BLEU metrics, because emoji can be fully encoded by the embedding layer after conversion, thus working in the variant-character-recognition task.

5.2 Analysis of model's efficiency

To quantitatively compare the computational efficiency of our SMTM with other baselines during the inference phase, we begin timing when the data entered the model and stopped once the loss values for each batch were obtained. We set each batch to 10 and calculated the average training time required for 10 batches, to facilitate model's efficiency comparison. Simulations were conducted on central processing units (CPUs), single graphics processing unit (S-GPU), and multi-GPUs (M-GPUs) to investigate the relationship between model's efficiency and simulation setup. The CPU simulation was conducted on a machine equipped with 16 Intel Xeon Silver 4110 CPUs (totaling eight cores), while the GPU simulation used an NVIDIA Tesla V100.

The time of the inference phase for the four models is shown in Fig. 3. Although the inference time of our SMTM is longer than that of the other baselines under CPU devices, the time under S-GPU and M-GPUs is only 0.14 and 0.05 min longer, respectively,

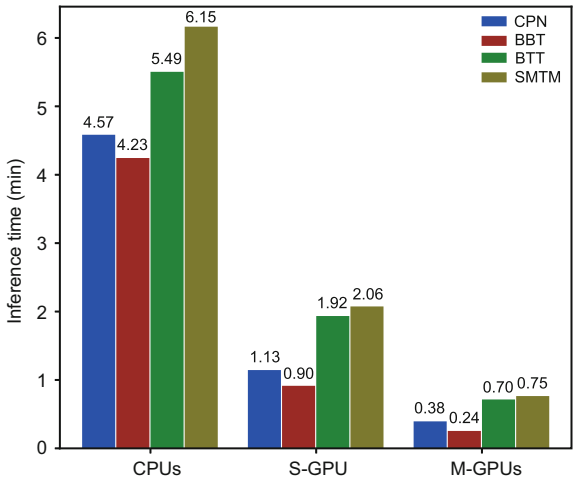

Fig. 3 Comparison of the results of model efficiency

than that of BTT on average. The reason is that for non-real-time tasks, our model has more learned parameters compared to the other three models, and both parameter learning and convergence processes consume time to improve the model's performance. Such time consumption is perfectly acceptable in the field of deep learning. In addition, the effect of different devices varies from one model to another, and when computing in parallel on multiple GPUs, our model consumes only 0.05 min more than the BTT model, while the performance remains optimal among all the models.

5.3 Error analysis

For a deeper understanding of the sources of errors and the types of errors corrected in our proposed SMTM, we randomly annotate 100 selected mistranslated Chinese variants. We categorize the error types into four categories: incomplete input, expression ambiguity, numerical errors, and font semantic errors, and the specific explanations of each error type are as follows:

1. Incomplete input. It happens when the input text of the model has Chinese simplified expressions or abbreviated expressions. This kind of expression usually requires the model to automatically complete the semantic information and recognize the logical relationship, or the translated information could be misrepresented.

2. Expression ambiguity. An emoticon exists in the input text of the model, and the meaning of the emoticon represents different types of semantic information according to various contextual information. This situation typically depends not only on the emoji encoding but also on the model's ability to accurately localize the semantics contained in the emoji based on contextual information.

3. Numerical errors. The presence of numbers in the input text, with the numbers being interspersed with Chinese numeric expressions, causes these errors. When such a situation exists even if the translation model can accurately recognize the Chinese variant characters, the model will show erroneous results in subsequent translations due to the inheritance of these numerical errors.

4. Font semantic errors. These errors are generally due to the presence of image-like textual representations in the dataset. When performing image recognition, encoding errors lead to deviations in the

semantics of the text, thus making it challenging for the model to provide a final accurate translation.

The analysis results for the error types of the annotated data are shown in Fig. 4. By comparing the error differences between our SMTM and the baseline model BTT, it can be seen that our model effectively corrects 28% of the errors from the baseline. In particular, the error type of incomplete input is reduced by 12 PPs, which indicates that our proposed model is capable of accurately encoding semantics based on contextual information when facing Chinese acronym expressions, proving that Pinyin embedding is effective in assisting BERT embedding to accurately learn the contextual semantics during the encoding process. Both the numbers of instances of expression ambiguity and font semantic errors decrease, demonstrating the effectiveness of the emoji enhancement and font embedding added to our model, which can accurately recognize expression and font semantics in Chinese. Numerical errors decrease by 11 PPs, reflecting the effectiveness of the model in recognizing numbers. In general, our model can significantly improve the ability to understand complex contexts, thus reducing the occurrence of errors when facing numbers, emojis, and fonts appearing in Chinese variant characters, improving the performance of the model and confirming its practical effect and application value in the Chinese variant-character-recognition task.

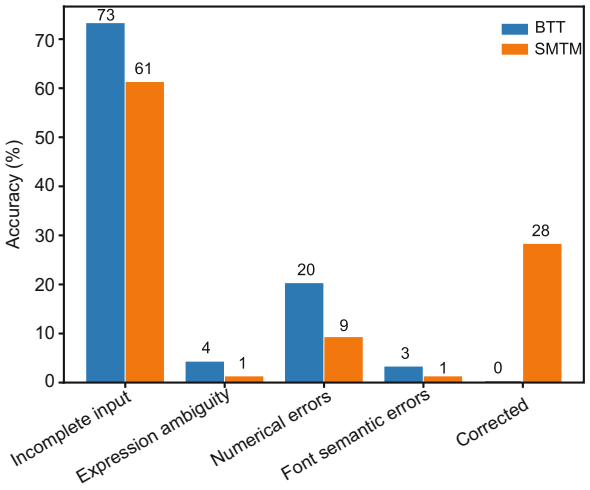

Fig. 4 Error analysis results with baseline BTT as the standard

5.4 Case study

To visually compare the quality of the models in translating Chinese variant characters, we conduct a case study and present it in Fig. 5, which contains the Chinese variant characters generated by the baseline model BTT and our proposed SMTM. The comparison shows that the BTT model is capable of accurately recognizing and translating Chinese variants of homophone and Pinyin substitution, while it is less effective for other types of variants. For Pinyin abbreviation, special symbol substitution, and Chinese–English mixture, BTT prefers to translate the variant characters into English. When facing similar character substitution and inversion of word order, although BTT can translate the correct Chinese characters, the semantics is not in conformity with the general expression. For the input text containing traditional character substitution, it

Variant mode	Text	BTT	SMTM
Homophone	刷单加威信	刷单加微信	刷单加微信
Pinyin substitution	刷单加wei信	刷单加微信	
Pinyin abbreviation	刷单加wx	刷单加wx	
Traditional character substitution	刷單加微信	刷unk加微信	
Similar character substitution	刷单加徽信	刷单加灰心	
Radical split	刷单力口微信	刷单利口微信	
Special symbol substitution	刷单+微信	刷单+微信	
Chinese–English mixture	刷单+V	刷单+V	
Inversion of word order	刷单加信微	刷单加欣慰	

Fig. 5 Case study

is treated as an unrecognized font in the translation process due to the limited encoding capability of the model. Our proposed SMTM accurately recognizes and translates the nine Chinese variant characters according to the contextual semantic expressions, which proves that the character's phonological and morphological features added in the encoding stage of our proposed model can accurately recognize various characters and Pinyin in the Chinese variant characters. In addition, the proposed shared-weight embedding mechanism can strengthen the connection between source and target characters and facilitate the generation of target sentences, thus improving the model's performance.

6 Conclusions

Current methods for Chinese character embedding typically use Chinese character tokens for processing. However, this unimodal approach faces challenges in capturing the phonological and morphological features of Chinese characters. In this paper, we present the SMTM. It mines the deep semantic features of Chinese variant text based on Chinese characters, Pinyin, and font images, and uses a shared-weight embedding mechanism to generate target sentences. In the simulations, we show that our model outperforms all compared models in the Chinese variant-character-recognition task. In the investigation of the Chinese variant-character-recognition task, precise classification and labeling of variant and standard characters are particularly crucial. The Uniform Content Label (UCL, GB/T 35304-2017) provides a standardized approach for classifying and labeling contents to improve the manageability and traceability of Chinese variant characters. UCL was originated by Professor Youping LI, an academician of the Chinese Academy of Engineering, and it can guarantee the security and reliability of data sources by using mechanisms such as dual signature, chain of accountability, and time–space self-consistency. Based on this, UCL could ensure the accuracy of deep learning-based Chinese variant-character-recognition tasks by discerning and eliminating forged and variant content through identification and filtering processes. In our future work, we will focus on the annotation and classification problem of Chinese variant characters using UCL to show better performance in the Chinese domain.

Contributors

Yuankang SUN designed the research and drafted the paper. Peng YANG oversaw and led the planning and execution of the study. Bing LI, Lexiang LI, and Dongmei YANG collected the information and revised the paper.

Conflict of interest

All the authors declare that they have no conflict of interest.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

Bao ZY, Li C, Wang R, 2020. Chunk-based Chinese spelling check with global optimization. Proc Findings of the Association for Computational Linguistics, p.2031-2040. https://doi.org/10.18653/v1/2020.findings-emnlp.184

Bryant C, Yuan Z, Qorib MR, et al., 2023. Grammatical error correction: a survey of the state of the art. *Comput Linguist*, 49(3):643-701. https://doi.org/10.1162/COLI_A_00478

Chang Y, Kong L, Jia KJ, et al., 2021. Chinese named entity recognition method based on BERT. Proc IEEE Int Conf on Data Science and Computer Application, p.294-299. https://doi.org/10.1109/ICDSCA53499.2021.9650256

Chen KH, Wang R, Utiyama M, et al., 2018. Syntax-directed attention for neural machine translation. Proc 32nd AAAI Conf on Artificial Intelligence, p.4792-4799. https://doi.org/10.1609/aaai.v32i1.11910

Cheng XY, Xu WD, Chen KL, et al., 2020. SpellGCN: incorporating phonological and visual similarities into language models for Chinese spelling check. Proc 58th Annual Meeting of the Association for Computational Linguistics, p.871-881. https://doi.org/10.18653/v1/2020.acl-main.81

Cho K, van Merriënboer B, Gulcehre C, et al., 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. Proc Conf on Empirical Methods in Natural Language Processing, p.1724-1734. https://doi.org/10.3115/v1/D14-1179

Choi H, Cho K, Bengio Y, 2018. Fine-grained attention mechanism for neural machine translation. *Neurocomputing*, 284:171-176. https://doi.org/10.1016/j.neucom.2018.01.007

Chollampatt S, Taghipour K, Ng HT, 2016. Neural network translation models for grammatical error correction. Proc 25th Int Joint Conf on Artificial Intelligence, p.2768-2774.

Cui YM, Che WX, Liu T, et al., 2021. Pre-training with whole word masking for Chinese BERT. *IEEE/ACM Trans Audio Speech Lang Process*, 29:3504-3514. https://doi.org/10.1109/TASLP.2021.3124365

Dabre R, Chu CH, Kunchukuttan A, 2021. A survey of multilingual neural machine translation. *ACM Comput Surv*, 53(5):99. https://doi.org/10.1145/3406095

Dai F, Cai Z, 2017. Glyph-aware embedding of Chinese characters. Proc 1st Workshop on Subword and Character Level Models in NLP, p.64-69. https://doi.org/10.18653/v1/W17-4109

Devlin J, Chang MW, Lee K, et al., 2019. BERT: pre-training of deep bidirectional Transformers for language understanding. Proc Conf of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, p.4171-4186. https://doi.org/10.18653/v1/N19-1423

Diao SZ, Bai JX, Song Y, et al., 2020. ZEN: pre-training Chinese text encoder enhanced by N-gram representations. Proc Findings of the Association for Computational Linguistics, p.4729-4740. https://doi.org/10.18653/v1/2020.findings-emnlp.425

Dubey A, Jauhri A, Pandey A, et al., 2024. The Llama 3 herd of models. https://arxiv.org/abs/2407.21783

Gehring J, Auli M, Grangier D, et al., 2017. Convolutional sequence to sequence learning. Proc 34th Int Conf on Machine Learning, p.1243-1252.

Hong YZ, Yu XG, He N, et al., 2019. FASPell: a fast, adaptable, simple, powerful Chinese spell checker based on DAE-decoder paradigm. Proc 5th Workshop on Noisy User-Generated Text, p.160-169. https://doi.org/10.18653/v1/D19-5522

Hu H, Richardson K, Xu L, et al., 2020. OCNLI: original Chinese natural language inference. Proc Findings of the Association for Computational Linguistics, p.3512-3526. https://doi.org/10.18653/v1/2020.findings-emnlp.314

Ji JS, Wang QL, Toutanova K, et al., 2017. A nested attention neural hybrid model for grammatical error correction. Proc 55th Annual Meeting of the Association for Computational Linguistics, p.753-762. https://doi.org/10.18653/v1/P17-1070

Jia C, Shi YF, Yang QR, et al., 2020. Entity enhanced BERT pre-training for Chinese NER. Proc Conf on Empirical Methods in Natural Language Processing, p.6384-6396. https://doi.org/10.18653/v1/2020.emnlp-main.518

Jia YZ, Xu XB, 2018. Chinese named entity recognition based on CNN-BiLSTM-CRF. Proc IEEE 9th Int Conf on Software Engineering and Service Science, p.1-4. https://doi.org/10.1109/ICSESS.2018.8663820

Jin H, Zhang ZB, Yuan PP, 2022. Improving Chinese word representation using four corners features. *IEEE Trans Big Data*, 8(4):982-993. https://doi.org/10.1109/TBDATA.2021.3106582

Li B, Yang P, Zhao HL, et al., 2023. Hierarchical sliding inference generator for question-driven abstractive answer summarization. *ACM Trans Inform Syst*, 41(1):7. https://doi.org/10.1145/351189

Li B, Yang P, Sun YK, et al., 2024. Advances and challenges in artificial intelligence text generation. *Front Inform Technol Electron Eng*, 25(1):64-83. https://doi.org/10.1631/FITEE.2300410

Li JT, Meng K, 2021. MFE-NER: multi-feature fusion embedding for Chinese named entity recognition. https://arxiv.org/abs/2109.07877

Li WG, Ramos RM, Brom PC, 2024. Threshold determination for Chinese character image processing in multimodal information fusion. Proc 28th Int Conf on Asian Language Processing, p.43-48. https://doi.org/10.1109/IALP63756.2024.10661155

Li WS, Wei YG, An D, et al., 2022. LSTM-TCN: dissolved oxygen prediction in aquaculture, based on combined model of long short-term memory network and temporal convolutional network. *Environ Sci Pollut Res*, 29(26):39545-39556. https://doi.org/10.1007/s11356-022-18914-8

Li XN, Yan H, Qiu XP, et al., 2020. FLAT: Chinese NER using flat-lattice Transformer. Proc 58th Annual Meeting of the Association for Computational Linguistics, p.6836-6842. https://doi.org/10.18653/v1/2020.acl-main.611

Liang ZY, Du JP, Li CY, 2020. Abstractive social media text summarization using selective reinforced Seq2Seq attention model. *Neurocomputing*, 410:432-440. https://doi.org/10.1016/j.neucom.2020.04.137

Liu J, Yang YH, Lv SQ, et al., 2019. Attention-based BiGRU-CNN for Chinese question classification. *J Amb Intell Human Comput*. https://doi.org/10.1007/s12652-019-01344-9

Liu JG, Xia CH, Li XJ, et al., 2020. A BERT-based ensemble model for Chinese news topic prediction. Proc 2nd Int Conf on Big Data Engineering, p.18-23. https://doi.org/10.1145/3404512.3404524

Liu SL, Yang T, Yue TC, et al., 2021. PLOME: pre-training with misspelled knowledge for Chinese spelling correction. Proc 59th Annual Meeting of the Association for Computational Linguistics and the 11th Int Joint Conf on Natural Language Processing, p.2991-3000. https://doi.org/10.18653/v1/2021.acl-long.233

Liu WJ, Zhou P, Wang ZR, et al., 2020. FastBERT: a self-distilling BERT with adaptive inference time. Proc 58th Annual Meeting of the Association for Computational Linguistics, p.6035-6044. https://doi.org/10.18653/v1/2020.acl-main.537

Liu Y, Lapata M, 2019. Hierarchical Transformers for multi-document summarization. Proc 57th Conf of the Association for Computational Linguistics, p.5070-5081. https://doi.org/10.18653/v1/P19-1500

Ma SM, Sun X, Lin JY, et al., 2018. Autoencoder as assistant supervisor: improving text representation for Chinese social media text summarization. Proc 56th Annual Meeting of the Association for Computational Linguistics, p.725-731. https://doi.org/10.18653/v1/P18-2115

Maruf S, Saleh F, Haffari G, 2022. A survey on document-level neural machine translation: methods and evaluation. *ACM Comput Surv*, 54(2):45. https://doi.org/10.1145/3441691

Meng FD, Zhang JC, 2019. DTMT: a novel deep transition architecture for neural machine translation. Proc 33rd AAAI Conf on Artificial Intelligence, p.224-231. https://doi.org/10.1609/aaai.v33i01.3301224

Meng FD, Lu ZD, Li H, et al., 2016. Interactive attention for neural machine translation. Proc 26th Int Conf on Computational Linguistics, p.2174-2185.

Meng YX, Wu W, Wang F, et al., 2019. Glyce: glyph-vectors for Chinese character representations. Proc 33rd Int Conf on Neural Information Processing Systems, Article 247.

Otter DW, Medina JR, Kalita JK, 2021. A survey of the usages of deep learning for natural language processing. *IEEE Trans Neur Netw Learn Syst*, 32(2):604-624. https://doi.org/10.1109/TNNLS.2020.2979670

Papineni K, Roukos S, Ward T, et al., 2002. BLUE: a method for automatic evaluation of machine translation. Proc 40th Annual Meeting of the Association for Computational Linguistics, p.311-318. https://doi.org/10.3115/1073083.1073135

Reimers N, Gurevych I, 2019. Sentence-BERT: sentence embeddings using siamese BERT-networks. Proc Conf on Empirical Methods in Natural Language Processing and the 9th Int Joint Conf on Natural Language Processing, p.3980-3990. https://doi.org/10.18653/v1/D19-1410

Shao YF, Geng ZC, Liu YT, et al., 2024. CPT: a pre-trained unbalanced transformer for both Chinese language understanding and generation. *Sci China Inform Sci*, 67(5):152102. https://doi.org/10.1007/s11432-021-3536-5

Sheng L, Xu ZX, Li XL, et al., 2023. EDMSpell: incorporating the error discriminator mechanism into Chinese spelling correction for the overcorrection problem. *J King Saud Univ-Comput Inform Sci*, 35(6):101573. https://doi.org/10.1016/j.jksuci.2023.101573

Soydaner D, 2022. Attention mechanism in neural networks: where it comes and where it goes. *Neur Comput Appl*, 34(16):13371-13385. https://doi.org/10.1007/s00521-022-07366-3

Stahlberg F, 2020. Neural machine translation: a review. *J Artif Intell Res*, 69:343-418. https://doi.org/10.1613/jair.1.12007

Sun Y, Wang SH, Li YK, et al., 2019. ERNIE: enhanced representation through knowledge integration. https://arxiv.org/abs/1904.09223

Sun ZJ, Li XY, Sun XF, et al., 2021. ChineseBERT: Chinese pretraining enhanced by glyph and pinyin information. Proc 59th Annual Meeting of the Association for Computational Linguistics and the 11th Int Joint Conf on Natural Language Processing, p.2065-2075. https://doi.org/10.18653/v1/2021.acl-long.161

Tao HQ, Tong SW, Zhao HK, et al., 2019. A radical-aware attention-based model for Chinese text classification. Proc 33rd AAAI Conf on Artificial Intelligence, p.5125-5132. https://doi.org/10.1609/aaai.v33i01.33015125

Vaswani A, Shazeer N, Parmar N, et al., 2017. Attention is all you need. Proc 31st Int Conf on Neural Information Processing Systems, p.6000-6010.

Wang DM, Song Y, Li J, et al., 2018. A hybrid approach to automatic corpus generation for Chinese spelling check. Proc Conf on Empirical Methods in Natural Language Processing, p.2517-2527. https://doi.org/10.18653/v1/D18-1273

Wang DM, Tay Y, Zhong L, 2019. Confusionset-guided pointer networks for Chinese spelling check. Proc 57th Annual Meeting of the Association for Computational Linguistics, p.5780-5785. https://doi.org/10.18653/v1/P19-1578

Wang YG, Cheng SB, Jiang LY, et al., 2017. Sogou neural machine translation systems for WMT17. Proc 2nd Conf on Machine Translation, p.410-415. https://doi.org/10.18653/v1/W17-4742

Weng RX, Yu H, Huang SJ, et al., 2020. Acquiring knowledge from pre-trained model to neural machine translation. Proc 34th AAAI Conf on Artificial Intelligence, p.9266-9273. https://doi.org/10.1609/aaai.v34i05.6465

Wu FZ, Liu JX, Wu CH, et al., 2019. Neural Chinese named entity recognition via CNN-LSTM-CRF and joint training with word segmentation. Proc World Wide Web Conf, p.3342-3348. https://doi.org/10.1145/3308558.3313743

Xie JB, Hou YJ, Wang YJ, et al., 2020. Chinese text classification based on attention mechanism and feature-enhanced fusion neural network. *Computing*, 102(3):683-700. https://doi.org/10.1007/s00607-019-00766-9

Xu HD, Li ZL, Zhou QY, et al., 2021. Read, listen, and see: leveraging multimodal information helps Chinese spell checking. Proc Findings of the Association for Computational Linguistics, p.716-728. https://doi.org/10.18653/v1/2021.findings-acl.64

Yan H, Deng BC, Li XN, et al., 2019. TENER: adapting Transformer encoder for named entity recognition. https://arxiv.org/abs/1911.04474

Yang A, Yang BS, Hui BY, et al., 2024. Qwen2 technical report. https://arxiv.org/abs/2407.10671

Yao YS, Huang Z, 2016. Bi-directional LSTM recurrent neural network for Chinese word segmentation. Proc 23rd Int Conf on Neural Information Processing, p.345-353. https://doi.org/10.1007/978-3-319-46681-1_42

Zhang B, Xiong DY, Xie J, et al., 2020. Neural machine translation with GRU-gated attention model. *IEEE Trans Neur Netw Learn Syst*, 31(11):4688-4698. https://doi.org/10.1109/TNNLS.2019.2957276

Zhang SH, Huang HR, Liu JC, et al., 2020. Spelling error correction with soft-masked BERT. Proc 58th Annual Meeting of the Association for Computational Linguistics, p.882-890. https://doi.org/10.18653/v1/2020.acl-main.82

Zhang Y, Liu YG, Zhu JJ, et al., 2019. Learning Chinese word embeddings from stroke, structure and pinyin of characters. Proc 28th ACM Int Conf on Information and Knowledge Management, p.1011-1020. https://doi.org/10.1145/3357384.3358005

Zhang YS, Zheng J, Jiang YR, et al., 2019. A text sentiment classification modeling method based on coordinated CNN-LSTM-attention model. *Chin J Electron*, 28(1):120-126. https://doi.org/10.1049/cje.2018.11.004

Zhao H, Cai D, Xin Y, et al., 2017. A hybrid model for Chinese spelling check. *ACM Trans Asian Low-Resour Lang Inform Process*, 16(3):21. https://doi.org/10.1145/3047405

Zhao S, Hu MH, Cai ZP, et al., 2023. Enhancing Chinese character representation with lattice-aligned attention. *IEEE Trans Neur Netw Learn Syst*, 34(7):3727-3736. https://doi.org/10.1109/TNNLS.2021.3114378

Zhou J, Cui GQ, Hu SD, et al., 2020. Graph neural networks: a review of methods and applications. *AI Open*, 1:57-81. https://doi.org/10.1016/j.aiopen.2021.01.001

Zhou SY, Xu S, Xu B, 2018. Multilingual end-to-end speech recognition with a single Transformer on low-resource languages. https://arxiv.org/abs/1806.05059v2

Zhuang H, Wang C, Li CL, et al., 2017. Natural language processing service based on stroke-level convolutional networks for Chinese text classification. Proc IEEE Int Conf on Web Services, p.404-411. https://doi.org/10.1109/ICWS.2017.46