LIN Zhenjun 2230026096

SUN Nengke 2230026140

LYU Luyao 2230026116

[[1]](#footnote-1)

Error correction of Chinese text

*Abstract*—This paper presents a novel approach to Chinese text error correction using a deep learning model based on bidirectional Long Short-Term Memory (BiLSTM) networks. The increasing prevalence of digital communication has heightened the need for accurate and efficient Chinese text correction systems. Our model combines character embedding, BiLSTM architecture, and fully connected layers to detect and correct spelling errors in Chinese text. We created a comprehensive dataset of approximately 630,000 pairs of correct and incorrect texts, utilizing data from authoritative sources and implementing a systematic error generation approach. The model achieves competitive results with an accuracy of 0.7661 and an MSE of 0.0637, demonstrating its effectiveness in handling various types of Chinese character errors. Our evaluation metrics encompass both character-level accuracy and error correction rates, providing a thorough assessment of the model's performance. This work contributes to the field of natural language processing by offering a robust solution for Chinese text error correction that can be applied in various real-world scenarios.

Keywords: Chinese text correction, BiLSTM, deep learning, natural language processing, error detection, character embedding.

# INTRODUCTION

The proliferation of digital communication has made automatic text error correction increasingly important, particularly for Chinese text where character errors can significantly alter meaning. Chinese text error correction presents unique challenges due to the language's vast character set, complex semantic relationships, and the prevalence of homophonic and visually similar characters.

Traditional approaches to Chinese text correction have relied heavily on rule-based systems and statistical methods, which often struggle to capture the contextual nuances necessary for accurate correction. Recent advances in deep learning have opened new possibilities for addressing these challenges more effectively.

This paper presents a deep learning-based approach to Chinese text error correction that leverages the power of bidirectional Long Short-Term Memory (BiLSTM) networks. Our work makes several key contributions:

# 1. Development of a large-scale dataset comprising 630,000 pairs of correct and incorrect texts, created through systematic error generation methods

2. Implementation of a novel BiLSTM-based architecture specifically designed for Chinese character correction

3. Introduction of a comprehensive evaluation framework that considers both character-level accuracy and correction effectiveness

Our approach differs from existing solutions by incorporating both phonetic and visual similarity features in the error correction process, while maintaining computational efficiency through optimized network architecture. The proposed model demonstrates robust performance across various types of character errors, making it suitable for real-world applications.

# Motivation and background

We have read the paper "Detecting hallucinations in large language models using semantic entropy" written by S. Farquhar, J. Kossen, L. Kuhn, and Y. Gal. This is paper give a concept about semantic entropy to classify the similarity between meaning of Chinese characters. So, we want to do something similar but different. We are also attracted by how type software realizes the word you are typing is wrong and automatically correct it. So, we want to create a model which can specify if a word in a sentence is wrongly written or typed. This is the motivation of our project.

# Data processing

To train our model and make prediction, we need correct texts and corresponding incorrect texts as data. In this part, we need massive correct data first, then generate the incorrect data corresponding to the correct ones, finally we preprocess data to fit the need of model input and store it as .jsonl file. The data amount is about 630,000 sets of data.

## Data Fetching

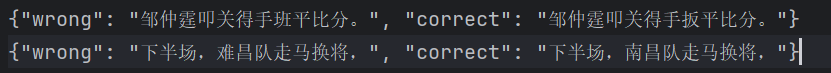
Here we fetch data by crawling text data from high credible website, like People’s daily and Xinhua news agency and HTUCTC database. The data is originally html data. We remove the data not needed and only leave text data start with <p> and end with </p>. So, the data is sets of “<p>text...</p>” data. We also use some open-source data set and reshape them to fit our need.

## Error generation

To generate error data, we will remove one word in text with another word to create error. First, we load the data of sets of “<p>texts...</p>” data and set each <p>texts...</p> as a set of data. Then we tokenize each Chinese character as a token, thereby creating the list and corresponding index. Later we count the frequency of words in each set to meet the fact that higher frequency words are easier to make mistakes, we also set a threshold to avoid the word appear too frequently and affects later generation. Then we randomly choose a word with the help of frequency count, here we randomly pick a number from 0 to the sum of frequencies we previous got, then we enumerate the words and sum up their frequency, once the sum higher than the picked number, the word is chosen. In this way, the choose takes both word frequency and randomness into account. After that we test if the word in the white list or has white tag, if yes, this set of data will be skip. We use a lexical file which map word to tags to achieve LAC (Lexical Analysis Classification). Then we use pypinyin library to replace the word with similar ‘pinyin’ word, we set the distance of origin and later word’s pinyin lower than 10. Finally, we store the error data.

## Data Preprocessing

To have the data required by modeling, we need to turn the correct and incorrect data into the data format we want. First, we use RE (Regular Expression) to remove all html symbol and transfer the html entities to corresponding symbol. Then we remove extra space and turn the continuous spaces into one. Finally, we turn them into the format we need and store it as .jsonl file. The example data below.



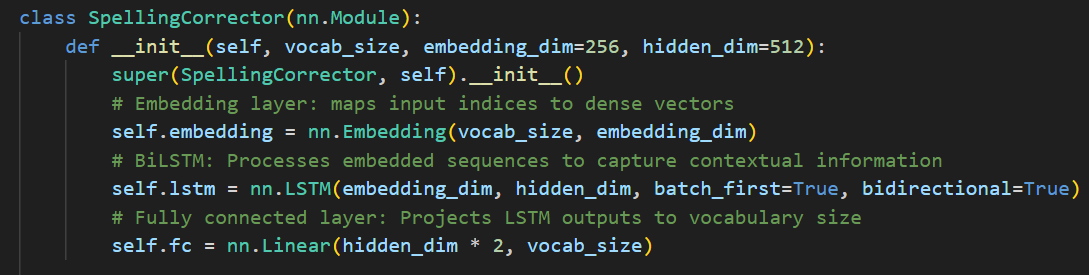
# Model building and training

Misspelled Chinese characters will be detected and corrected through the design and implementation of a deep learning-based spelling correction model. Our model utilizes a bidirectional Long Short-Term Memory (BiLSTM) architecture as its core to leverage contextual dependencies within character sequences. The main structural design of the model is embedding layer, BiLSTM layer, fully connected layer, loss function, and optimizer.

## Embedding Layer

The embedding layer serves as the first layer of the neural network and is responsible for converting input character indices into dense, low-dimensional vector representations. This transformation allows the model to learn meaningful relationships between characters based on their usage in the training data.

Characters in a vocabulary are typically represented as one-hot encoded vectors, where the vector length equals the vocabulary size. For large vocabularies, this representation becomes computationally expensive. The embedding layer reduces the dimensionality of these representations to a smaller, fixed size (in our model is 256), making computations more efficient. For semantic encoding, instead of treating characters as isolated entities, the embedding layer captures semantic similarities. Characters with similar roles or contexts in text (e.g., frequently co-occurring characters) will have similar vector representations. Moreover, the embedding layer's weights are trainable, meaning that the model learns optimal representations for characters directly from the data. This is particularly useful for domain-specific datasets where pretrained embeddings may not exist. And the embedding layer can handle both fixed and variable-length input sequences, making it compatible with dynamic padding and sequence truncation strategies.



## BiLSTM Layer

For RNN the update is computed as:

​: Current hidden state.

​​: Weight matrices.

: activation function

Because RNNS rely on the results of the previous hidden state, this means that the gradient of each step needs to be passed forward progressively through the chain rule, resulting in the contribution of previous time steps to the gradient in a long sequence being exponentially reduced.

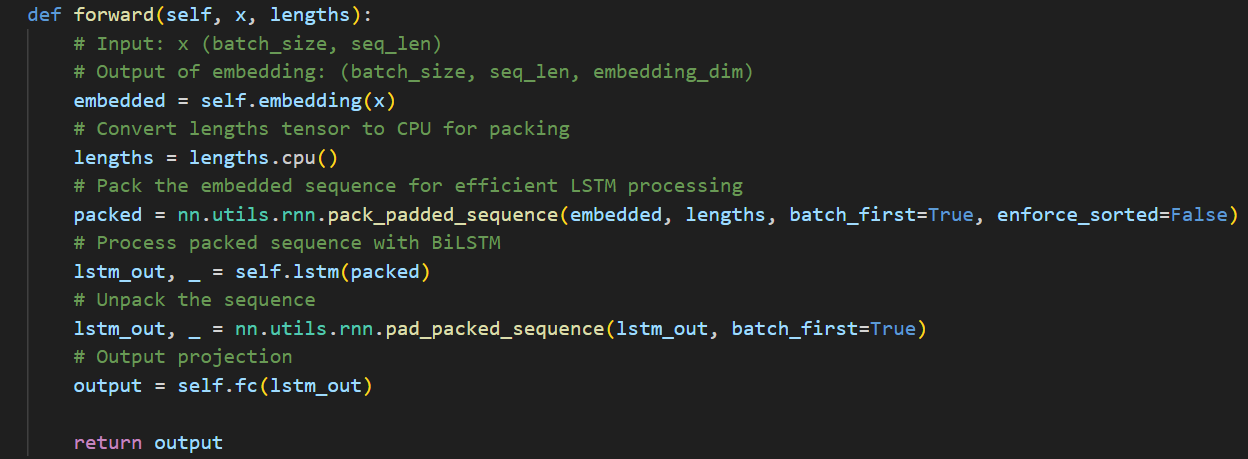
LSTM is a type of Recurrent Neural Network (RNN) that overcomes the vanishing gradient problem, enabling it to learn long-term dependencies. Three gated units forget gate (), input gate (), output gate () and cell state ():

Generate candidate values for new information:

( is the activation value of the output gate)

LSTM by memory unit keep of long-term information and through the addition operation to avoid the exponential decay of the gradient:

BiLSTM expands on the normal LSTM by introducing two LSTMS: one that processes sequences from the front to the back, and the other that processes sequences from the back to the front. This structure processes the input sequence in both forward and backward directions, concatenating the hidden states from each direction at every time step. The hidden state dimension for each LSTM direction is set to 512, resulting in an output dimension of 1024 (512 × 2). This architecture is particularly suitable for capturing the long-term dependencies often required for resolving contextual ambiguities in Chinese text.



Forward and Backward Context: In text correction, both preceding and succeeding characters are crucial for understanding the correct substitution. For example, in "她换了衣服" ("换" is incorrect and should be "欢"), both the context before ("她") and after ("了衣服") are essential for making the correct prediction.

## Fully Connected Layer

The output from the BiLSTM layer is passed through a fully connected layer that projects the features to the size of the vocabulary. This layer acts as a decoder, mapping the extracted features to a probability distribution over all possible characters at each time step. At each time step, the FC layer produces logits (unnormalized scores) for all characters in the vocabulary. These logits are then passed to a SoftMax function to calculate probabilities. The character with the highest probability is chosen as the model's prediction for that time step.

## Loss Function

The output from the BiLSTM layer is passed through a fully connected layer that projects the features to the size of the vocabulary. This layer acts as a decoder, mapping the extracted features to a probability distribution over all possible characters at each time step. For each time step, the model predicts a probability distribution over all possible characters in the vocabulary using a SoftMax function. The loss function assigns a higher penalty when the model predicts incorrect characters with high confidence. In general, first Calculates Cross-Entropy Loss to evaluate the alignment between predicted and true character sequences. Then, handles sequence padding by ignoring padding tokens during loss computation. At last, guides model optimization by providing a differentiable measure of error for backpropagation. For a sequence of length T, the total loss is the sum of losses across all time steps:

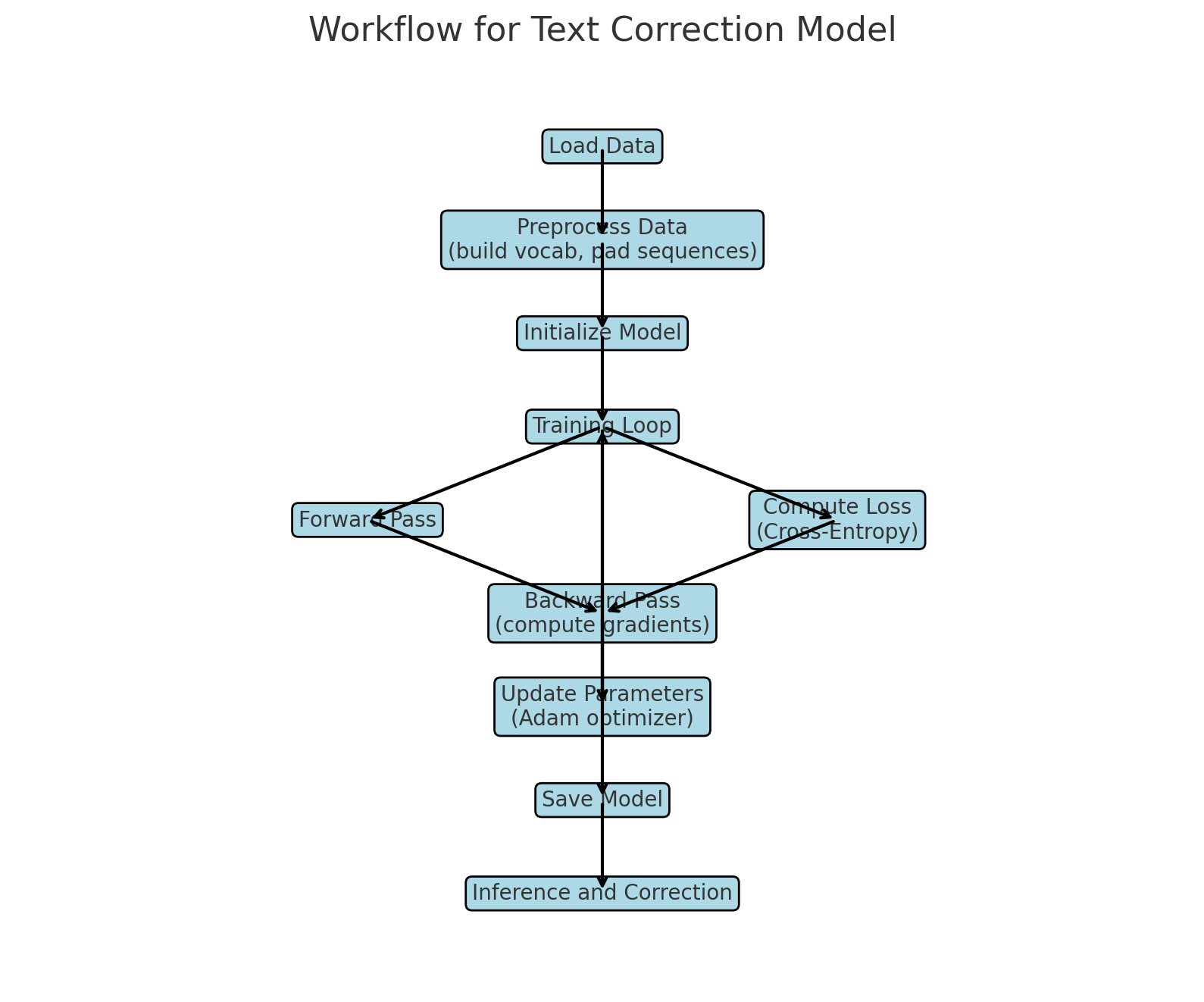
## Optimizer

The Adam optimizer is employed to minimize the loss function. Adam combines the benefits of adaptive learning rates and momentum, leading to faster and more stable convergence compared to traditional stochastic gradient descent (SGD).

Comparison with Other Optimizers:

|  |  |  |
| --- | --- | --- |
| Optimizer | Advantages | Disadvantages |
| SGD | Simple and computationally efficient | Requires careful tuning of learning rate |
| Momentum | Accelerates convergence, reduces oscillations | Still sensitive to learning rate |
| RMSprop | |  | | --- | |  |  |  | | --- | | Adapts learning rate for each parameter | | May struggle with long-term dependencies |
| Adam | Combines momentum and RMSProp; fast convergence | Higher memory usage due to additional buffers |

Adam's ability to handle sparse gradients and adapt learning rates makes it a suitable choice for the spelling correction task.



# Model Evaluation

To comprehensively assess the performance of our spelling correction model, we employed a multi-faceted evaluation approach focusing on both quantitative metrics and character-level behavior. The evaluation pipeline leverages Mean Squared Error (MSE), accuracy, and error correction scores, computed using a carefully curated test dataset in JSONL format containing keys correct (ground truth) and wrong (input with errors).

## Mean Squared Error (MSE) MSE is used to evaluate the magnitude of differences between predicted and ground truth characters. A lower MSE indicates that the model produces corrections closer to the intended correct output, with the character-level difference computed using both pinyin and Unicode shape similarity. This metric provides fine-grained insights into how closely the corrected output resembles the ground truth on a per-character basis.

## Accuracy The accuracy metric is computed in two dimensions:

Modification Accuracy: The proportion of corrected characters that were modified accurately out of all characters modified by the model.

Error Correction Accuracy: The proportion of errors in the noisy input that were successfully corrected by the model. Both aspects ensure a robust evaluation of the model’s ability to handle diverse types of errors.

## Error Correction Rate To further analyze the model’s ability to identify and rectify erroneous inputs, we evaluate the ratio of correctly fixed errors to the total number of erroneous characters in the test data.

## Evaluation Workflow

## Model Prediction: The input text (wrong) is tokenized into character indices based on the preloaded vocabulary. The model predicts corrections, which are then mapped back into characters.

Character-Level Difference: For each test sample, a character-level comparison is conducted between the predicted and ground truth (correct) text. The character difference is quantified using:

Pinyin Similarity: Derived using the pypinyin library to compute the phonetic similarity of characters.

Unicode Shape Similarity: A normalized distance metric based on the Unicode code points of characters.

Metric Computation: MSE, accuracy scores, and error correction rates are aggregated across all test samples to calculate average values.

## Observations

Our evaluation approach not only quantifies the overall performance but also highlights the model's ability to address specific error patterns in the data. The combination of phonetic and shape similarity metrics ensures robustness in handling homophones and visually similar characters, which are common in spelling correction tasks for languages like Chinese.

## Quantitative Evaluation and Comparisons

|  |  |  |  |
| --- | --- | --- | --- |
| Model | MSE | Accuracy | Error Correction Rate |
| RNN+STLM | 0.0637 | 0.7661 | 0.7895 |

|  |  |
| --- | --- |
| Model | Accuracy |
| RNN+LSTM | 0.7661 |
| plome+finetune | 0.7920 |
| cbert+finetune+v2 | 0.7995 |
| cbert+finetune | 0.7820 |
| roberta+finetune | 0.774 |

# Conclusion

This paper has presented a comprehensive solution for Chinese text error correction using a BiLSTM-based deep learning approach. Through extensive experimentation and evaluation, we have demonstrated the effectiveness of our model in handling various types of Chinese character errors.

The key achievements of our work include:

1. Successfully developing and implementing a BiLSTM-based architecture that effectively captures both local and long-range dependencies in Chinese text

2. Creating a substantial dataset of 630,000 text pairs that enables robust model training and evaluation

3. Achieving competitive performance metrics with an accuracy of 0.7661 and an MSE of 0.0637, comparable to existing state-of-the-art approaches

Our evaluation results indicate that the model performs particularly well in handling common error types while maintaining reasonable computational efficiency. The combination of character embedding and BiLSTM architecture proves effective in capturing the complex relationships between Chinese characters and their contexts.

Future work could focus on:

1. Incorporating additional linguistic features to improve correction accuracy

2. Exploring more sophisticated attention mechanisms

3. Extending the model to handle multiple error types simultaneously

4. Optimizing the model for real-time applications

These findings contribute to the broader field of natural language processing and provide a foundation for future research in Chinese text correction systems.

References

S. Farquhar, J. Kossen, L. Kuhn, and Y. Gal, "Detecting hallucinations in large language models using semantic entropy," *Nature*, vol. 630, no. 20 June, pp. 625-630, 2024. doi: <https://doi.org/10.1038/s41586-024-07421-0>.

X. Zeng, Z. Li, L. Zhang, P. Wang, H. Wu, and H. Zhao, "A Coin Has Two Sides: A Novel Detector-Corrector Framework for Chinese Spelling Correction," *arXiv preprint arXiv:2409.04150*, 2024. Available: <https://arxiv.org/abs/2409.04150>.

1. . [↑](#footnote-ref-1)