

Article

KSEC: A Knowledge-Enhanced Approach for Variable-Length Chinese Spelling Correction

Jiahao Wang ¹, Guimin Huang ^{1,2}, Yabing Wang ^{1,*}, Qingkai Guo ¹, Yiqun Li ¹ and Nanxiao Deng ¹

¹ School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin 541004, China; wangjh@mails.guet.edu.cn (J.W.); sendhuang@126.com (G.H.); guoqkai@126.com (Q.G.); 22031002019@mails.guet.edu.cn (Y.L.); nanxiao.deng@mails.guet.edu.cn (N.D.)
² Guangxi Key Laboratory of Image and Graphic Intelligent Processing, Guilin 541004, China
* Correspondence: wangyabing819@guet.edu.cn

Abstract

Chinese Spelling Correction (CSC) is a fundamental task in Natural Language Processing (NLP) aimed at identifying and correcting character errors in Chinese texts. It significantly enhances text readability and semantic accuracy. Most deep learning-based CSC methods focus on isometric correction, ensuring identical lengths for input and output sequences. However, they struggle with variable-length errors like splitting errors—where a single character is incorrectly divided into two (e.g., splitting “明” into “日” and “月”). These errors are challenging because they disrupt token alignment, preventing standard sequence-labeling models from mapping inputs to outputs effectively. To overcome this limitation, we propose KSEC (Knowledge-enhanced Splitting Error Corrector), a novel framework tailored for variable-length corrections. KSEC automatically constructs a splitting character knowledge base from public corpora to provide factual validation for correction outcomes. Furthermore, we design a variable-length architecture integrating an attention mechanism and introduce an alignment-aware loss function that optimizes sequence-to-sequence token mapping. Extensive experiments on standard CSC and CSEC benchmarks demonstrate that KSEC achieves state-of-the-art performance among lightweight models of similar size and outperforms existing methods across multiple evaluation metrics.

Keywords: Chinese spelling correction; attention mechanism; deep learning; natural language processing

1. Introduction

Chinese Spelling Correction (CSC) automatically detects and corrects spelling errors in text. It is an essential preprocessing step for various downstream applications, including search engines [1], optical character recognition (OCR) [2], automatic speech recognition (ASR) [3], and large language models (LLMs) [4]. However, traditional CSC models carry a strict limitation: they rely heavily on an equal-length assumption, meaning the input and output sequences must contain the exact same number of characters. Beyond traditional spelling errors, splitting errors are highly prevalent in real-world scenarios such as OCR outputs, handwritten inputs, and online conversations, especially for learners of Chinese as a second language. This learner-centric perspective aligns with broader educational research emphasizing that effective automated assistance must account for the specific error patterns and cognitive contexts of target user groups [5]. These errors occur when a single Chinese character is incorrectly broken down into multiple separate characters.



Academic Editor: Arkaitz Zubiaga

Received: 16 June 2026

Revised: 19 July 2026

Accepted: 20 July 2026

Published: 22 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

For example, the character “好” (good) might be split into “女” (woman) and “子” (child). Such errors severely damage semantic coherence. Because splitting errors result in mismatched input and output lengths, they directly violate the equal-length assumption of conventional CSC systems, creating a major performance bottleneck.

To address this challenge, Liang et al. [6] formally introduced the Chinese Splitting Error Correction (CSEC) task. We note that the same acronym CSEC has been independently used by Huang et al. [7] to refer to a Chinese Semantic Error Correction dataset for written correction; to avoid any ambiguity, we clarify that throughout this paper CSEC exclusively denotes the Chinese Splitting Error Correction task and benchmark of Liang et al. They constructed a dual-domain dataset covering both news and social media content, establishing a benchmark for future model evaluation. Figure 1 illustrates a real-world example of these errors, highlighting misspelled tokens and their correct form, where misspelled characters are marked in red and their corrections in green.

Error Type	Sentence
Phonetic error	记得按时迟(吃)早饭。 Remember to late (eat) breakfast on time.
Visual error	夏天的天气非常谈(炎)热。 The weather in summer is very talk(hot).
Common error	我每天六天(点)起床。 I get up at six days (o'clock) every day.
Splitting error	他的爱好是运云力(动)。 His hobby is cloud power(sports).

Figure 1. Examples of spelling and splitting errors. Incorrect characters are highlighted in red, and correct characters are highlighted in green.

Early CSC methods primarily focused on combining phonetic and glyph features to correct spelling errors [8–10]. These approaches developed specialized encoders to capture both character pronunciations and visual shapes [8,9]. To handle multiple concurrent errors and prevent over-correction, subsequent methods introduced multi-round and error-guided mechanisms [11,12]. These frameworks isolate errors through iterative processing [11] or targeted position masking [12]. KSEC is related to edit-based grammatical error correction methods, such as edit-operation tagging and copy-and-generate models, which also handle variable-length edits. However, KSEC is designed for Chinese spelling correction rather than general grammatical error correction, and its novelty lies in combining source-aligned deletion with a splitting-character knowledge base for Chinese character-level merging. In the field of Chinese grammar error correction (CGEC), edit-based grammar error correction methods, such as edit-operation tagging [13], and copy-and-generate mechanisms [14–16] also handle variable-length edits. Since the two tasks involve different types of errors, the underlying correction logic is also different. Generic variable-length GEC mechanisms do not explicitly model the structural relation between Chinese character components and their merged character. GECToR lacks graphic decomposition knowledge for multi-to-one merges, while copy-and-generate models do not explicitly ground such merges in component structure. Recent approaches use large language models to extract contextual information, outperforming traditional confusion sets [17]. However, general-domain models and GPT-4 [18] often show poor cross-domain performance in specialized fields [19]. To improve model interpretability, some advanced frameworks align neural networks with structured knowledge associations [18]. Others shift from tradi-

tional sequence labeling to a sentence-rewriting paradigm to mimic human-like cognitive patterns [19].

Although these three paradigms—ranging from multi-modal feature encoding [8,9] and iterative positioning [11,12] to advanced LLM-based context modeling [17,19–21]—have advanced the field, they all operate under a rigid, equal-length substitution assumption. These existing models lack the mechanism to process sequence length variations, rendering them ineffective against splitting errors where a single character is fragmented into multiple tokens. Therefore, a knowledge-enhanced splitting-error corrector model is urgently required to break the rigid length constraints of traditional networks by leveraging structured radical and glyph knowledge, thereby enabling the framework to dynamically reconstruct split characters without sacrificing standard spelling correction precision.

In this paper, we propose the Knowledge-enhanced Splitting Error Corrector (KSEC) to simultaneously correct Chinese spelling and splitting errors. Evaluated on the public benchmarks, KSEC achieves highly competitive performance. KSEC operates as a text-level corrector: it takes an existing character sequence as input and outputs the corrected sequence, regardless of whether the text originates from keyboard input, OCR, handwriting, or social media decomposition. Consequently, OCR engine accuracy and input method specifics are upstream concerns outside the scope of this work. Our main contributions are as follows:

1. We construct a splitting knowledge base using radical and structural features to provide explicit linguistic constraints. Based on this, we design a soft pre-detection module that combines the knowledge base with MacBERT to score candidate splitting spans. This soft mechanism guides the corrector effectively while preventing the accidental removal of valid characters.
2. We introduce a variable-length correction framework using source-length-aligned labels and a specialized deletion token [DEL]. This design enables the model to alter the output length and fix splitting errors in a single forward pass, completely removing the traditional equal-length constraint.
3. We propose a comprehensive loss function that jointly optimizes character prediction, length prediction, and knowledge-guided merging. This cooperative training objective balances standard spelling correction and sequence length changes, significantly reducing both over-correction and under-correction.

2. Methodology

In this section, we detail KSEC, a unified framework that jointly handles conventional Chinese spelling errors and the comparatively under-studied splitting errors.

2.1. Splitting Knowledge Base

We construct a Splitting Knowledge Base (SKB) designed to map sequences of two or three adjacent source characters—the visual sub-components generated by a splitting error—back to their intended unified compound characters. Formally, this knowledge base is defined as shown in Equation (1).

$$K(x_i x_{i+1} \dots x_{i+k-1}) = c, k \in \{2, 3\}, \quad (1)$$

where x_i denotes a fragmented component sequence (e.g., {王, 求}) and c represents the target merged character (e.g., {球}).

The SKB is systematically assembled from two complementary sources. An example of the knowledge base is shown in Figure 2. Crucially, the SKB automatically integrates

static structural character data with dynamic error patterns mined from public corpora via alignment models.

ID Chinese character Confusion set of Chinese character	Split character	Confusion set of split character
3198 时 5495 肘 3333 村 2561 付 1397 吋 802 付	3189 日	3288 曰 3299 月 4680 目 3190 旦 4635 白
	2189 寸	2127 守 2191 寺
1225 励 5490 肋 1221 助 1237 勋	1326 厉	674 万 6815 迈 1325 历
	1213 力	1143 刀 1145 刃 711 为

Figure 2. Examples of the splitting knowledge base.

The Chinese splitting errors frequently exhibit top-to-bottom and multi-component patterns (e.g., “田心” forms “思” and the three-component “立日心” forms “意”). To capture these diverse variations, our model extracts splitting confusions directly from an aligned error-correction corpus. Specifically, we apply a character-level diff algorithm to align each source–target training pair, isolating edit spans where two or three contiguous source characters collapse into a single target character. Spans where all constituent units constitute valid Chinese characters are retained as candidate entries. To mitigate alignment noise, a mined entry is incorporated only when its corpus frequency is at least 2.

These corpus-derived entries are subsequently integrated with the structural dictionary. The mined entries are then merged with the structural table, with corpus evidence taking precedence where the two disagree, since it reflects errors that actually occur in practice. The structural part of the SKB is built from the compound-character table of the General Standard Chinese Characters list, which provides about 2043 left–right structural entries. After merging these entries with corpus-mined splitting patterns, the final SKB contains about 2469 entries covering component sequences of lengths two and three. The SKB serves dual downstream objectives: it facilitates the pre-detection module (Section 2.2) by suggesting candidate splitting spans, and it guides the confusion loss (Section 2.3.4) to regularize the corrector toward knowledge-supported merges. Importantly, since a significant portion of this knowledge base is constructed from observed real-world errors, its coverage adaptively aligns with the error distribution of the target task rather than being restricted to a fixed, a priori configuration.

2.2. Knowledge-Based Splitting Error Pre-Detection

To identify potential splitting errors within the input sequence, the model first scans the sentence using sliding windows of lengths 2 and 3 to search for matches in the splitting knowledge base. While character n -gram models are computationally efficient, they cannot detect splitting errors where components form valid high-probability sequences, nor can they perform the variable-length merging operations required for correction. These structural limitations motivate our knowledge-based approach, which grounds candidate detection in a structured splitting knowledge base rather than surface-level co-occurrence statistics. However, the candidate spans identified during this initial scan frequently overlap and conflict (for instance, a single character cannot belong to two distinct merging operations simultaneously). To resolve this, we employ a greedy conflict resolution strategy. We assign priorities based on the candidate spans’ co-occurrence frequencies in the corpus,

prioritizing high-frequency candidates and immediately discarding any other options that intersect with their positions. After this filtering process, the remaining candidate spans are mutually exclusive. We define this conflict-free candidate set as a collection of triplets:

$$C = \{(i_p, j_p, c_p)\}_{p=1}^m \quad (2)$$

where $[i_p, j_p]$ denotes the start and end indices of the potential splitting error in the original sentence, and c_p is the correct merged character provided by the knowledge base.

To prevent the model from blindly applying knowledge base rules out of context, which could lead to over-correction, we further introduce a secondary confirmation mechanism powered by a masked language model (MLM). Specifically, we replace the selected candidate spans with mask tokens and observe the predictions of MLM based on the surrounding context. In our implementation, we set $k = 3$. If the target character c_p appears among the top 3 characters with the highest predicted probabilities, we consider the splitting error valid within the current context. This semantic confidence evaluation process can be expressed by the following equation:

$$\text{conf}(i_p, j_p, c_p) = 1 \left[c_p \in \text{Topk} \left(P_{\text{MLM}} \left(\bullet \mid X_{[i_p:j_p] \leftarrow [\text{MASK}]} \right) \right) \right] \quad (3)$$

Finally, this verified candidate set C helps us determine the length range of the corrected sentence. Assuming the original sentence length is n and the set contains $|C|$ independent potential errors, the sentence would shorten by at most $|C|$ characters in the extreme case where all candidates require merging. Accordingly, we can establish a precise search interval for the length prediction head:

$$R_{\text{len}} = [n - |C|, n] \quad (4)$$

Defining this interval not only provides the corrector with reasonable physical boundaries for sentence contraction but also significantly reduces the computational complexity of the subsequent decoding process.

It is important to emphasize that the SKB serves as a soft guidance mechanism rather than a hard constraint. When the pre-detection module proposes no candidate, the predicted length offset defaults to zero and KSEC reduces exactly to a standard length-preserving CSC model (Section 2.3.2). An undetected out-of-vocabulary (OOV) splitting therefore does not cause catastrophic failure—the sentence is simply handled as in conventional CSC, and the correction/detection paths remain fully functional.

2.3. Knowledge-Enhanced Splitting Error Corrector

Although a splitting error changes the sentence length, the change is always a contraction that can be expressed as deletions over the source sequence. We therefore abandon the conventional masked language model with a separate length-changing decoder and instead build the corrector on an encoder-only pretrained backbone, aligning every prediction to a source position and realizing length change through an explicit deletion mechanism. This makes the entire model a single-pass, end-to-end trainable network, while two knowledge-injection channels carry the external splitting knowledge into the encoder. We initialize the backbone from MacBERT and extend its vocabulary with two special tokens, [LENGTH] and [DEL], whose roles are detailed below. The overall architecture of KSEC is shown in Figure 3.

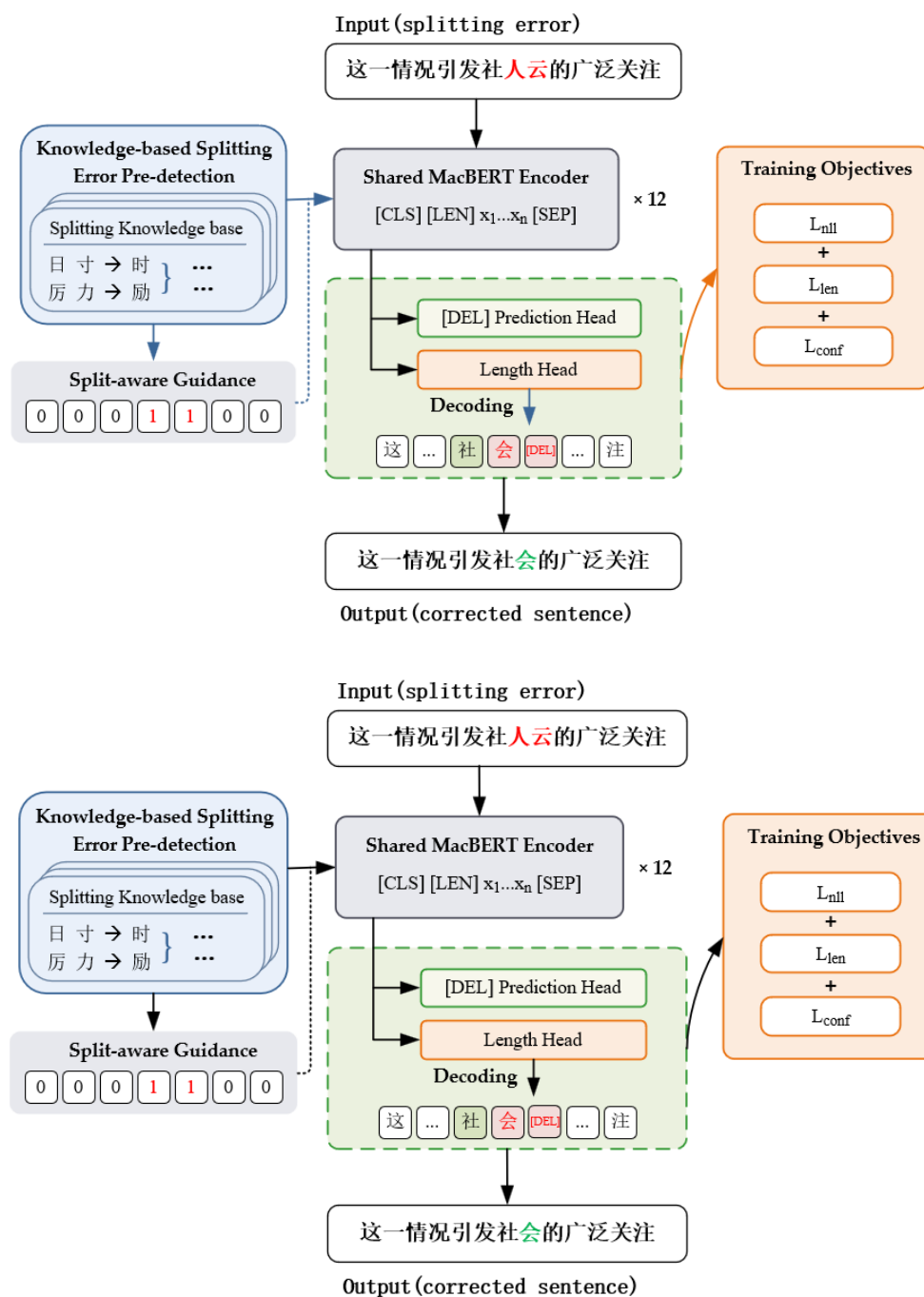


Figure 3. Overview of the proposed KSEC framework. Chinese characters in the figure serve as exemplary sentences and knowledge base entries to demonstrate the splitting error correction process.

2.3.1. Splitting Error Aware Encoder

We treat the correction process as a marking task for each position of source alignment. Given a source–target sentence pair, we align them using a character-level difference algorithm and assign a label to each source position. For regular positions that require no modification or involve equal-length substitutions, we directly label them with the correct target characters. Conversely, for a splitting span where k source characters are merged into a single target character, we assign the merged character to the start position and label the remaining $k - 1$ positions with a deletion token, [DEL]. Consequently, the label sequence strictly aligns with the source sequence in length. During inference, the model predicts the [DEL] tokens for redundant components, which are then trivially removed in a post-processing step to naturally realize character merging. The total length of the

predicted target sentence is simply the original sequence length n minus the number of predicted [DEL] tokens. The equation is as follows:

$$y_i = \begin{cases} c & \text{if } i = h \\ [\text{DEL}] & \text{if } h < i < h + k \\ t_i & \text{otherwise} \end{cases} \quad (5)$$

where y_i denotes the target label assigned to the i -th position of the source sequence; h and k represent the start index and the length of the potential splitting span, respectively; and c is the target compound character mapped by the knowledge base, while t_i refers to the correct target character for regular positions (i.e., copy or equal-length substitution). The special deletion token [DEL] is utilized to pad the redundant source positions during the merging operation.

To accommodate this process, the model's input sequence is formatted as [CLS] [LENGTH] $x_1 x_2 \dots x_n$ [SEP]. The backbone network encodes this sequence in a single forward pass. Subsequently, a masked language model classification head outputs the target-character probability distribution at each source position. At this stage, we inject a split-aware signal: rather than modifying the self-attention matrix with explicit masks, we leverage the encoder's token-type input to directly mark positions that fall within candidate splitting spans (using gold spans during training and pre-detected spans during inference). Through the standard token-type embedding mechanism, this mark translates into an additive bias on the hidden states of the corresponding positions, guiding the model to actively attend to potential splitting components. This implementation is computationally lightweight, requires no modification to the underlying attention computation, and is trained end-to-end jointly with the rest of the network. The key calculation processes are shown in Equation (6):

$$h_i^{(0)} = E_{\text{tok}}(x_i) + E_{\text{pos}}(i) + E_{\text{type}}(s_i), \quad s_i \in \{0, 1\} \quad (6)$$

In the input representation, $h_i^{(0)}$ is defined as the initial hidden state vector of the i -th token fed into the 0-th layer of the backbone network. This vector is constructed by summing the token embedding E_{tok} of the input character x_i , the absolute position embedding E_{pos} , and the token-type embedding E_{type} . The binary split-aware indicator signal is denoted by s_i : $s_i = 1$ if the i -th position falls inside a candidate splitting span, and $s_i = 0$ otherwise.

2.3.2. Scoped Length Prediction

Since parallel non-autoregressive generation cannot rely on an end-of-sequence token to terminate decoding, the degree of sequence contraction must be modeled explicitly. To this end, we prepend a [LENGTH] token to the input sequence and project its contextualized hidden state through a dedicated linear classification head. Rather than predicting the absolute target length, this head estimates the deletion count—defined as the offset between the source and target lengths—over a discrete set of possible offset classes. This predicted offset corresponds precisely to the total number of [DEL] tokens emitted for the sentence, thereby establishing a direct coupling with the source-aligned deletion mechanism detailed in Section 2.3.1 and providing sentence-level supervision for sequence contraction. In parallel, the pre-detection module described in Section 2.2 introduces a complementary prior by defining a plausible contraction range $[n - |C|, n]$. The inclusion of weak candidates within m deliberately loosens this range's boundaries, ensuring that this prior guides rather than rigidly restricts the final merging decision. In the absence of

any detected splitting anomalies, the target offset defaults to zero, whereby the framework seamlessly reduces to a standard and length-preserving CSC model.

The [LENGTH] head is not redundant with the per-position classification head. While the classification head makes independent local decisions at each source position, it can be noisy or inconsistent during early training stages. The [LENGTH] head provides complementary sentence-level supervision that regularizes the total contraction degree and prevents the model from arbitrarily shortening the sentence. This global signal stabilizes training and narrows the decoding search space, working in tandem with the local [DEL] predictions.

2.3.3. Knowledge-Based Decoding

At each source position, the model outputs the token with the highest predicted probability from the character classification head. Specifically, positions predicted as [DEL] are directly dropped, naturally realizing a many-to-one character merge. Positions predicted as other regular characters undergo in-place substitution to address conventional spelling correction, while those matching the original source characters remain unchanged.

With this design, the number of dropped positions directly determines the final length of the generated sentence. Consequently, a single decoding pass can seamlessly accommodate both length-preserving character substitutions and length-reducing merge operations, eliminating the need for additional complex decoding stages. Notably, because the character classification head is explicitly optimized via a tailored confusion loss during training, it is heavily biased towards outputting the target characters from the knowledge base at identified splitting positions. This ensures that the merging decisions made during the reasoning process are based on external split knowledge rather than relying solely on the prior knowledge of the language model.

$$P(y_i | X) = \text{softmax}(W_o h_i^{(L)} + b_o), y_i \in V \cup \{[\text{DEL}]\} \quad (7)$$

At the output classification stage, $h_i^{(L)}$ denotes the deep contextualized hidden representation after passing through the L-layer transformer encoder; W_o and b_o are the trainable weight matrix and bias vector of the MLM classification head, respectively; and V defines the standard target vocabulary.

During inference, the two heads may produce inconsistent counts: the length head predicts an offset of k deletions, while the classification head outputs d [DEL] tokens (d not equal k). We resolve this by treating the classification head as the primary authority, since it accesses fine-grained, position-level context. The length head serves as a secondary guard: if d is substantially larger than k , we flag a possible anomaly and filter out low-probability [DEL] tokens using a confidence threshold. Thus, the length head acts as a soft prior rather than a hard constraint. In practice, such conflicts seldom occur after training convergence, because Equation (12) jointly optimizes both heads over shared encoder states, encouraging consistent behavior.

2.3.4. Loss Function

We decompose the overall objective function into three distinct components. These components jointly supervise conventional text correction, the degree of sentence contraction, and knowledge-grounded character merging.

The first component is the negative log-likelihood loss of the character classification head over all source sequence positions. The calculation equation for this part of the loss function is shown in Equation (8). We treat the special deletion token [DEL] directly as an ordinary class within the vocabulary. Therefore, this single loss term simultaneously

supervises both standard character substitutions and the deletion operations required for character merging.

$$L_{nll} = -\sum_i \log P(y_i | X) \quad (8)$$

The second component is dedicated to supervising the scoped length prediction head, and the equation is defined as shown in Equation (9). We compute the cross-entropy loss over the discrete classes of deletion counts, which indicate the length offset between the source and target sequences. During this process, we strictly restrict the prediction range to the pre-defined contraction interval. This constraint prevents the model from arbitrarily altering the sentence length during generation.

$$L_{len} = -\log P(L = L_k) \quad (9)$$

The specific class probabilities are obtained by applying a softmax function over the offset logits as shown in Equation (10):

$$P(L = L_k) = \text{softmax}(W \cdot h_{[\text{LENGTH}]} + b)_k \quad (10)$$

In the above equation, L_{len} denotes the scoped length loss, and L_k represents the ground-truth discrete length offset class. The vector $h_{[\text{LENGTH}]}$ is the contextualized hidden state of the [LENGTH] token at the beginning of the input sequence, as extracted by the encoder. Furthermore, W and b are the learnable weight matrix and bias vector of the dedicated length classification head, respectively. $P(L = L_k)$ is the probability that the model predicts the length offset to be exactly L_k . This value is obtained by applying the softmax operation to the output vector of the classification head and extracting the k -th component.

The third component is the confusion loss as shown in Equation (11). This mechanism is designed to explicitly inject the prior rules from the splitting knowledge base into the parameters of the corrector. Specifically, at each detected splitting-head position, this loss calculates the negative log-likelihood of the target character prescribed by the knowledge base. This operation effectively steers the character classification head toward merges supported by external knowledge. If the current sentence contains no splitting errors, this term provides no gradient contribution.

$$L_{conf} = -\frac{1}{N} \sum_{p \in S} \log P(y_p = c_p | X) \quad (11)$$

In this equation, the S represents the set of pre-detected splitting-head positions in the current sentence. The variable N is the total number of positions in this set, serving as a normalization factor. Additionally, $p \in S$ is the specific position index within the set, while c_p is the correct merged target character explicitly specified by the knowledge base at position p .

The final optimization objective is the weighted sum of the three aforementioned loss terms as shown in Equation (12). We introduce two constant hyperparameters θ and λ to balance the length loss, the confusion loss, and the foundational correction loss.

$$L = L_{nll} + \theta \times L_{len} + \lambda \times L_{conf} \quad (12)$$

In this way, we have achieved a good connection between the error detection and correction processes, and are able to conduct end-to-end joint optimization in the same shared pre-trained backbone network.

3. Experiments

This section presents extensive experiments to evaluate the performance of the KSEC. We conduct comprehensive tests on public datasets across both the CSC and CSEC tasks. Empirical comparisons with competitive baseline models show that KSEC performs competitively on both tasks.

3.1. Experimental Setup

3.1.1. Dataset

We used the benchmark datasets from the CSC and CSEC tasks. The overall scale of datasets is presented in Table 1 below. The training data consists of length-aligned spelling-error datasets from SIGHAN14 [22], SIGHAN15 [23], and CSCD [24], supplemented by pseudo-training data from Wang271K [25] generated via ASR and OCR confusion sets. We jointly train the model on these combined training sets and evaluate its performance on the SIGHAN15 and CSCD test sets. In all of these standard CSC datasets, the source and target sentences are of equal length, ensuring a direct mapping for character correction.

Table 1. The statistics of all the datasets used by KSEC.

Task		Name	#Sent	Avg Len	#Errors	#Err Sent
CSC	Train	SIGHAN14	3437	49.56	5136	3358
		SIGHAN15	2339	31.32	3048	2273
		CSCD	30,000	57.39	15,143	13,789
		Wang271K	271,329	42.56	381,962	271,009
		Total	307,105	44	405,289	290,429
	Test	SIGHAN15	1100	30.65	703	541
		CSCD	5000	57.63	2528	2303
		Total	6100	52.76	3231	2844
CSEC	Train	News	7869	30.68	8021	4968
		Social	6330	28.84	6172	3795
		Total	14,199	29.86	14,193	8763
	Test	News	1174	30.89	1214	775
		Social	1108	28.61	1089	671
		Total	2282	29.78	2303	1446

For splitting errors, we adopt the CSEC dataset introduced by Liang et al. during the training and testing phases. This dataset automatically inserts machine-generated splitting characters into social media texts and news texts with a probability of 0.1 using a Chinese split dictionary, effectively simulating the over-splitting phenomena commonly observed in online chatting and OCR processing. The overall scale of the datasets is summarized in Table 1. Where #Sent, #Errors, and #Err Sent denote the number of sentences, the number of errors, and the number of sentences containing errors, respectively.

3.1.2. Metrics

CSC evaluation typically relies on two primary criteria: sentence-level and character-level. At the sentence level, a prediction is considered fully correct only if all erroneous characters are amended without altering any correct text. Specifically, detection success requires predicted edit positions to match the ground truth exactly, while correction success additionally requires the final output sentence to match the target sequence. At the character level, identical metrics are computed across individual edit operations. In order to conduct a rigorous comparison with the baseline model, we adopted the sentence-level

metrics provided in [26] for the SIGHAN15 test set, and we reported character-level metrics for the CSCD test set in accordance with the standard protocol [18].

For CSEC, splitting errors alter sentence lengths; we rely on character-level alignment to extract edit spans rather than applying rigid position-wise comparisons. This alignment ensures that detection and correction metrics remain precisely defined when input and output lengths diverge. We use ChERRANT [27] as the evaluation criterion and report the P, R, and F1 values of the model in the news and social domains.

3.1.3. Baseline Models

ReLM [21] treats CSC as a sentence-rewriting task based on the semantics of the original sentence. ReLM is evaluated as zero-shot learning on the SIGHAN dataset and achieved state-of-the-art (SOTA) performance (F1 = 57.0). We use the splitting-error dataset for training and keep its original training strategy.

EGCM [12] adds the guidance strategy in the process of correcting errors, and only makes a prediction at the wrong position, while the correct position is not changed. This method has previously achieved SOTA results in correction quality.

SCOPE [8] adopts an encoder and two decoders in the model structure, and introduces glyph information and pinyin information in the middle layer. SCOPE achieves high-quality spelling-error correction through pronunciation prediction and multiple iterations.

ECSpell [9] adopts the error-consistent masking strategy for pre-training and constructs domain-specific datasets.

PLOME [28] implements a confusion-set-based masking strategy to jointly learn semantic knowledge and spelling errors. Moreover, the phonological and visual similarity knowledge of characters are also applied to the process of modeling.

REALISE [29] utilizes multimodal information for CSC, mixing the semantic, phonetic and graphic information of inputs to predict the correct output.

SpellGCN [30] incorporates visual and phonological similarity knowledge through a graph convolutional network.

BERT [10] is a very representative classical model that is suitable for most NLP tasks. We fine-tune the BERT model using the splitting-error dataset and show its performance on SIGHAN and splitting-error test set.

SoMu [6] employs a soft mask strategy to identify potential error marks and uses a transformer layer for error correction, achieving the correction of spelling errors and splitting errors.

Baichuan [31] is a representative large-scale Chinese–English bilingual foundation model trained on an extensive high-quality corpus.

Qwen3 [32] is a highly competitive, open-access large language model that demonstrates strong performance across various natural language processing benchmarks. In this work, we deploy it in a zero-shot setting to investigate the instruction-following ability of advanced LLMs when processing unstructured splitting errors.

GPT-4 [18] is an advanced closed-source large language model renowned for its exceptional general reasoning and cross-domain capabilities.

CSRP [33] has proposed a three-stage training framework, achieving the latest SOTA performance on the CSCD and NACGEC benchmarks.

For all zero-shot LLM baselines, we use the same Chinese instruction prompt: “你是一名中文文本纠错专家。请纠正输入句子中的拼写错误和拆分字错误。只纠正错误，不要改写 语 义或润色； 若无错误则原样输出。只输出纠正后的句子。” The input sentence is then provided as the user message, and no few-shot examples are used. The LoRA-tuned

Qwen3-8B baseline uses the same instruction prompt, so the comparison between zero-shot and fine-tuned Qwen3 keeps the prompt fixed.

3.2. Implementation Details

We build KSEC on the publicly available MacBERT-base backbone, which comprises 12 transformer encoder layers with 12 attention heads per layer, 768-dimensional hidden states, and a vocabulary that we extend with the two special tokens [LENGTH] and [DEL]. Rather than training from scratch, we fine-tune from the pretrained weights, which we found essential for data-efficient convergence. The loss weights θ and λ , which balance the scoped length-prediction loss and the knowledge-grounded confusion loss against the correction loss, are both set to 2. This setting is chosen because the correction loss is computed over all source positions, while the length loss and confusion loss are much sparser. We further verify this choice through a full sensitivity analysis over $\theta, \lambda \in \{0.5, 1, 2, 4\}$, which shows that $\theta = \lambda = 2$ achieves the best CSEC-Overall F0.5 while maintaining stable CSCD performance. We optimize with AdamW ($\beta = (0.9, 0.999)$, $\epsilon = 1 \times 10^{-6}$) using a batch size of 64 and a peak learning rate of 5×10^{-5} with 6% linear warmup followed by linear decay, and we cap the input length at 128 tokens. Because the CSC data is much larger than the split data, we oversample the CSEC split pairs by a factor of 3 so that the variable-length correction path stays well supervised. All experiments are performed on GeForce RTX 5090.

3.3. Main Results

This section presents three distinct comparisons. The first two evaluate conventional, length-preserving spelling correction on the SIGHAN15 and CSCD benchmarks. The third tests splitting-error correction on the CSEC news and social datasets. We compare our model against representative CSC systems from the past five years, several recent large language models, and the strongest dedicated baselines.

We start our analysis from the results of SIGHAN15 as shown in Table 2. Since this benchmark contains exclusively equal-length errors and lacks splitting cases, it serves as a sanity check. Our goal here is to confirm that the added splitting mechanisms do not compromise KSEC's performance on standard correction tasks.

Table 2. The evaluation results of CSC on the SIGHAN15 test set. The best results in each column are highlighted in bold.

Model	Detection Level			Correction Level		
	P	R	F ₁	P	R	F ₁
SpellGCN	74.8	80.7	77.7	72.1	77.7	75.9
REALISE	77.3	81.3	79.3	75.9	79.9	77.8
PLOME	77.4	81.5	79.4	75.3	79.3	77.2
ECSpell	76.4	79.9	78.1	74.4	77.9	76.1
EGCM	82.7	77.6	80	80.6	74.7	77.5
BERT-FT	75.3	79.3	77.2	73.1	76.9	74.9
SCOPE	79.5	84.3	81.8	78.6	83.4	80.9
ReLM	—	—	—	77.3	84.1	80.6
SoMu	78.3	82.9	80.5	76.5	81	78.9
KSEC (ours)	83.5	81.4	82.4	82.2	80.1	81.1

The experimental results shown in Table 2 indicate that KSEC achieved the best correction F1 (81.1) and detection F1 (82.4) among all baseline models. The margin over the previous best, SCOPE, is small in correction F1 (81.1 against 80.9), but the two models

reach it in different ways. SCOPE has higher recall (83.4 against 80.1), while KSEC has clearly higher precision (82.2 against 78.6, a 3.6-point gap). Against the knowledge-driven baselines REALISE and PLOME the lead is larger, about 3.3 and 3.9 F1 points. The main reason is that KSEC only edits positions when the encoder and the knowledge channel are consistent, so it changes fewer characters that are already correct.

To further demonstrate the capability of the KSEC model in CSC tasks, we conducted tests on a larger-scale CSCD dataset. The experimental results are shown in Table 3. The results show that although KSEC has only 126 M of parameters, it exceeds even the large model with 8B parameters in the CSC benchmark test.

Table 3. The evaluation results of CSC on the CSCD test set. The best results in each column are highlighted in bold.

Model	Detection Level			Correction Level		
	P	R	F ₁	P	R	F ₁
Qwen3 (4B)	30.1	23.2	26.2	27.2	21	23.7
Qwen3 (8B)	38.6	31.9	34.9	35.2	29	31.8
PGT (BERT)	-	-	-	-	-	48.6
GPT4	64.2	53.1	58.1	58.9	50.3	54.2
MDCSpell + ARM	-	-	-	-	-	48.9
CSRP (4B)	74	56.1	63.8	67.7	51.4	58.4
KSEC (Ours)	76.2	60.2	67.2	69.7	55.1	61.5

The comparison results between KSEC and the baseline LLMs are shown in Table 3. Experiments show that KSEC reaches a correction F1 of 61.5, while Qwen3-4B and Qwen3-8B reach only 23.7 and 31.8, a gap of roughly 30 to 38 F1 points. Even GPT-4 stops at 54.2, so KSEC is 7.3 points higher, a 13.5% relative gain. Against CSRP, the strongest CSC baseline, KSEC leads by 3.1 points in correction F1 (61.5 against 58.4) and 3.4 points in detection F1 (67.2 against 63.8). KSEC is more precise (69.7 against 67.7) and also has higher recall (55.1 against 51.4). The conclusion from SIGHAN15 therefore carries over to this much noisier data, where KSEC corrects the real errors while leaving correct text untouched.

In the third comparative experiment, we evaluated the split word error correction capability of the KSEC model through the CSEC benchmark. We evaluated the CSEC news and social test sets and reported the results in these two areas respectively. To offer a more direct comparison with generative correction methods, we additionally incorporate two fine-tuned generative baselines. Both models are trained on the same training data as KSEC, featuring identical data sources and the same oversampling strategy for CSEC split pairs, and are evaluated via the same metric pipeline. Specifically, we fine-tune BART-large-Chinese as an encoder–decoder generation baseline, and implement LoRA finetuning on Qwen3-8B using the same instruction prompt as in the zero-shot setting. Compared with the CSC test set, CSEC is more prone to overcorrection. Therefore, we use F0.5 as the evaluation metric instead of F1 to highlight the importance of accuracy. The detailed evaluation results are shown in Table 4.

Table 4 reports the experimental results, where the performance gap between KSEC and the general zero-shot models is the most pronounced among the three experiments. On the news and social datasets, KSEC achieves F0.5 scores of 77.0 and 69.6, respectively. In contrast, zero-shot large language models (LLMs) underperform significantly. GPT-4o scores 56.4 on news and 36.8 on social, which lags behind KSEC by 20.6 and 32.8 points, respectively. GPT-4 exhibits even lower performance on the social domain, scoring only 14.5. These results indicate that current general-purpose LLMs cannot effectively resolve splitting correction tasks out of the box.

Table 4. The evaluation results of CSEC on the CSEC test set. The best results in each column are highlighted in bold.

Model	News			Social		
	P	R	F _{0.5}	P	R	F _{0.5}
BART-large-FT	72	75.7	72.7	64.3	68.6	65.1
Baichuan (7B)	27.2	36.5	28.7	13.6	24.9	15
Qwen3 (4B)	18.5	17.9	18.2	7	6.4	6.7
Qwen3 (8B)	23.3	25.4	24.3	7	8.5	7.7
Qwen3-FT (8B)	74	75.9	74.3	69	67.9	68.7
GPT3.5	21.2	24.7	21.8	8.3	12.5	8.9
GPT4	54.6	56.3	54.9	18.4	7.8	14.5
GPT4o	56.2	57.5	56.4	35.5	43.1	36.8
SoMu	96	96.1	96	60.2	95.1	65
KSEC (Ours)	76.4	79.6	77	68.4	74.8	69.6

After fine-tuning, however, Qwen3-FT (8B) achieves substantially stronger performance, reaching F0.5 scores of 74.3 on CSEC-news and 68.7 on CSEC-social. This demonstrates that generative LLMs can benefit considerably from task-specific supervision. Nevertheless, Qwen3 (8B)-FT still falls behind KSEC by 2.7 points on news and by 0.9 points on social, indicating that KSEC remains competitive even compared with a fine-tuned generative model.

A more meaningful comparison can also be made with SoMu. On the clean news domain, SoMu delivers a strong performance of 96.0, outperforming KSEC by 19.0 points. However, its performance drops sharply to 65.0 on the noisy social domain, falling 4.6 points below KSEC. This performance degradation is explained by the precision and recall of SoMu on the social dataset: a high recall of 95.1 coupled with a low precision of 60.2 indicates that SoMu tends to merge text aggressively, leading to over-correction in informal contexts. Conversely, KSEC maintains a better balance, achieving a precision of 68.4 and a recall of 74.8. The results on CSEC-News suggest that SoMu is especially effective on clean, dictionary-generated split patterns because it applies merging more aggressively. In contrast, KSEC uses knowledge- and context-constrained merging, which may reduce recall on clean text but helps avoid over-correction in noisier social text.

To examine the stability of KSEC under different random seeds, we repeated the experiments three times and report the mean score with 95% confidence intervals. Figure 4 summarizes the results on SIGHAN15, CSCD, and CSEC test sets.

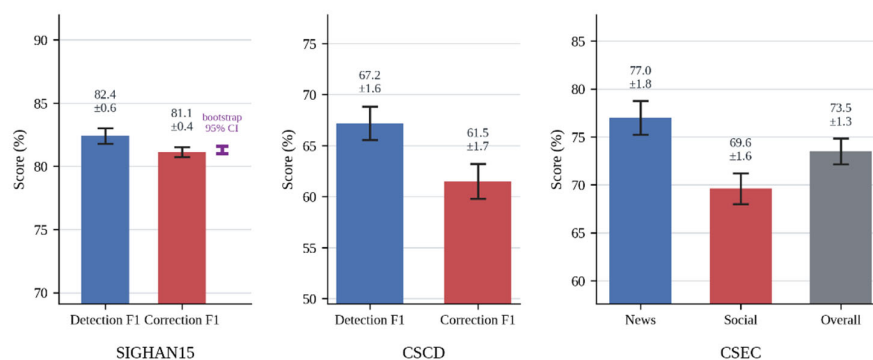
**Figure 4.** Multi-seed stability of KSEC.

Figure 4 shows that KSEC has different levels of stability across the three benchmarks. On SIGHAN15, the correction F1 is highly stable across the three runs, with almost no visible variation among the seed points. The bootstrap interval is wider than the seed-level in-

terval, which suggests that the uncertainty mainly comes from the test-set sampling rather than random initialization. This result indicates that KSEC behaves consistently on the standard sentence-level CSC benchmark. On CSCD, both detection and correction F1 show moderate variation. The confidence intervals are wider than those on SIGHAN15, but the three seed points remain close to each other. This suggests that KSEC is still reasonably stable on the larger character-level CSC benchmark.

The largest variation appears on CSEC, especially in the CSEC-news, CSEC-social, and overall F0.5 scores. This is expected because CSEC involves variable-length splitting errors, which are more sensitive to prediction boundaries and domain differences. Even with this larger uncertainty, the seed points follow the same overall trend across domains. The news domain obtains the highest score, the social domain is lower, and the overall score lies between them. This pattern shows that KSEC is more reliable on cleaner news text, while noisy social text remains more challenging.

To further demonstrate the reasonableness of the selection of loss weights θ and λ , we carried out experiments on different value combinations of θ and λ while keeping all other training settings unchanged. Figure 5 shows the overall results of different values for the CSCD and CSEC test sets.

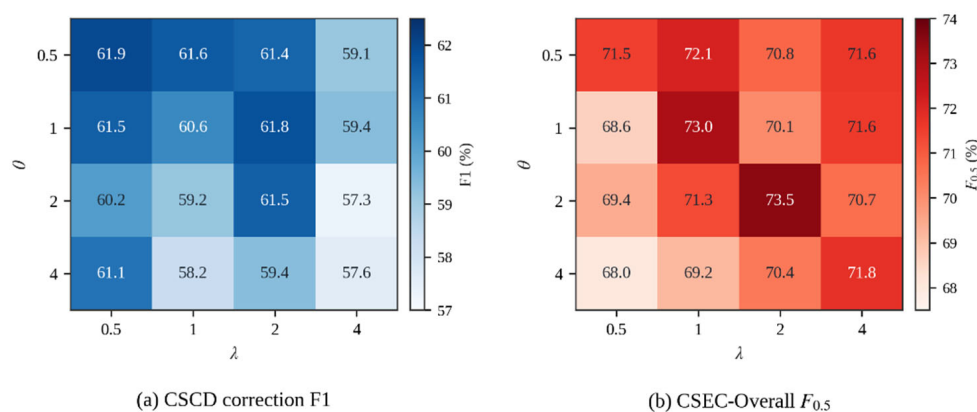


Figure 5. F-score sensitivity of KSEC under different loss weight settings.

On CSCD, the performance remains relatively stable across the grid, with character-level correction F1 ranging from 57.3 to 61.9. Although setting both θ and λ to 2 is not the absolute best setting on CSCD, it achieves a competitive F1 of 61.5, while providing the best CSEC-overall $F_{0.5}$. This indicates that our values of θ and λ offers the most favorable trade-off for the target splitting-error correction task without substantially sacrificing conventional CSC performance.

3.4. Ablation Study

Although the main results show the effectiveness of KSEC, the contribution of each component still needs to be examined separately. We therefore conduct two types of ablation experiments. The first removes one component at a time, denoted as w/o SKB, w/o DEL, and w/o KB-Loss. The second keeps only one component and removes the other two, denoted as only SKB, only DEL, and only KB-Loss. This design allows us to test not only whether each component is useful, but also whether any single component can work independently. Given that these components may affect performance differently across different tasks, we evaluate the ablation variants on two distinct benchmarks. Tables 5 and 6 report the ablation results for spelling correction and splitting correction, respectively.

Table 5. The ablation experiment results for CSC on the CSCD test set. The best results in each column are highlighted in bold.

Model	Detection Level			Correction Level		
	P	R	F ₁	P	R	F ₁
w/o SKB	69.4	59.2	63.9	63.1	53.8	58.1
w/o DEL	70.5	56.9	63	64.6	52.2	57.8
w/o KB-Loss	66.3	59.8	62.9	62.2	56.1	59
only SKB	67.5	55	60.6	61.9	50.4	55.6
only DEL	72.1	57.1	63.7	66.2	52.5	58.5
only KB-Loss	71.2	52.9	60.7	63.2	47	53.9
full KSEC	76.2	60.2	67.2	69.7	55.1	61.5

Table 6. The ablation experiment results for CSEC on the CSEC test set. The best results in each column are highlighted in bold.

Model	News			Social		
	P	R	F _{0.5}	P	R	F _{0.5}
w/o SKB	15.8	15.5	15.7	26.2	26.4	26.2
w/o DEL	-	-	-	-	-	-
w/o KB-Loss	69.5	73.2	70.2	62.3	68.6	63.5
only SKB	-	-	-	-	-	-
only DEL	5.7	5.4	5.6	14.4	13.7	14.2
only KB-Loss	-	-	-	-	-	-
full KSEC	76.4	79.6	77	68.4	74.8	69.6

To analyze the framework, we evaluate the following three reduced configurations:

w/o SKB: In this setting, the knowledge-based pre-detection module is removed. The model does not receive any candidate text spans proposed by the splitting knowledge base.

w/o DEL: In this setting, the deletion mechanism is disabled. As a result, the model cannot utilize the deletion marker [DEL] to shorten candidate sentences during the decoding phase.

w/o KB-Loss: In this setting, the specialized confusion loss function is excluded from the training objective, removing the targeted optimization for knowledge-based character merging.

only SKB: In this setting, only the knowledge-based pre-detection module is retained. The model receives candidate split spans from the knowledge base, but it cannot shorten the sentence or use the confusion loss.

only DEL: In this setting, only the deletion mechanism is retained. The model can use the [DEL] marker to shorten the sentence, but it has no knowledge-based span guidance or confusion-loss supervision.

only KB-Loss: In this setting, only the confusion loss is retained. The model keeps the training signal for knowledge-based merging, but it has no pre-detected candidate spans and cannot explicitly delete characters during decoding.

Table 5 shows the ablation results for spelling correction task. The full KSEC model achieves the best overall performance, with 67.2 detection F1 and 61.5 correction F1. Removing one component leads to moderate drops in correction F1: 3.4 points without SKB, 3.7 points without DEL, and 2.5 points without KB-Loss. This shows that none of the components harm conventional CSC, even though they are mainly designed for splitting errors. The only-component results further show that a single component is not sufficient. Among the three settings, only DEL performs best, reaching 58.5 correction F1, but it still

remains below the full model. Only SKB and only KB-Loss are weaker. These results suggest that the components are complementary even on equal-length correction, while their main value becomes clearer on the splitting correction task.

As shown in Table 6, the ablation results on the splitting correction task reveal a more distinct trend, demonstrating that all three components are indispensable. Removing SKB causes a large performance drop, reducing F0.5 from 77.0 to 15.7 on news and from 69.6 to 26.2 on social. This indicates that the knowledge-based pre-detection module is crucial for locating candidate split spans. Removing DEL leads to complete failure, because the model can no longer shorten the input sentence. This confirms that the deletion mechanism is necessary for performing merge operations. Removing KB-Loss causes a smaller but clear decline, from 77.0 to 70.2 on CSEC-news and from 69.6 to 63.5 on CSEC-social, showing that it mainly improves the accuracy of the merged character. The only-component settings confirm this conclusion. Only SKB and only KB-Loss cannot solve the splitting correction task because neither setting can complete the length-changing merge operation. Only DEL produces a small number of correct predictions, but its F0.5 remains very low, with 5.6 on CSEC-news and 14.2 on CSEC-social. Therefore, CSEC cannot be solved by any single component alone. SKB locates where a merge may occur, DEL executes the length change, and KB-Loss helps choose the correct merged character. The full model performs best because these three parts work together.

3.5. Case Study

To evaluate the practical performance of KSEC and the baseline LLMs, we conduct a case analysis using two representative examples. We compare KSEC with a SOTA baseline and two general-purpose LLMs. The first example represents a conventional substitution error of identical length, while the second illustrates a splitting error. These two examples represent the two correction capabilities evaluated in the previous section. For each case, we present the source input, the gold target, and the outputs generated by each model. In Figure 6, correct corrections are denoted with checkmarks, whereas incorrect outputs are marked with cross symbols.

As shown in Figure 6, the model output clearly compares the correction capabilities among different systems. In the first case, there is a phonetic similarity error (“时物” should be “食物”). It can be seen from the results that Qwen3-4B has made excessive deletions to the original sentence, while Qwen3-8B has made incorrect modifications. Only CSRP and KSEC have been modified completely correctly. In the second case, there was a splitting error involved (“人云” should be merged into “会”). Among all the results, the output of Qwen3-4B demonstrates the limitation of free generation, that is, over-correction will introduce new errors. CSRP failed to detect the splitting error in the original sentence, resulting in the correction failure. In contrast, KSEC precisely identified split characters and successfully merged “人云” into the correct “会”, maintaining a perfect alignment with the gold standard goals. These results show that KSEC can handle both conventional spelling errors and splitting errors in the tested cases.

To further investigate the internal mechanisms of these models, we analyzed the attention heatmaps of the models during the reasoning process, as shown in Figures 7 and 8. For traditional CSC tasks (Figure 7), KSEC’s attention matrix (a) shows a highly concentrated alignment of erroneous positions, while general LLMs (b, c) and baseline CSRP (d) display overly dispersed or biased attention weights, resulting in missed detections or overcorrections. For the more challenging CSEC tasks (Figure 8), KSEC (a) captures the remote dependencies of the segmented characters very well, effectively focusing its attention on the problematic range to guide the merge operation. On the contrary, the baseline

models (b, c, d) either exhibit the standard diagonal attention pattern and fail to identify structural splits, or fail to allocate sufficient attention to the wrong areas.

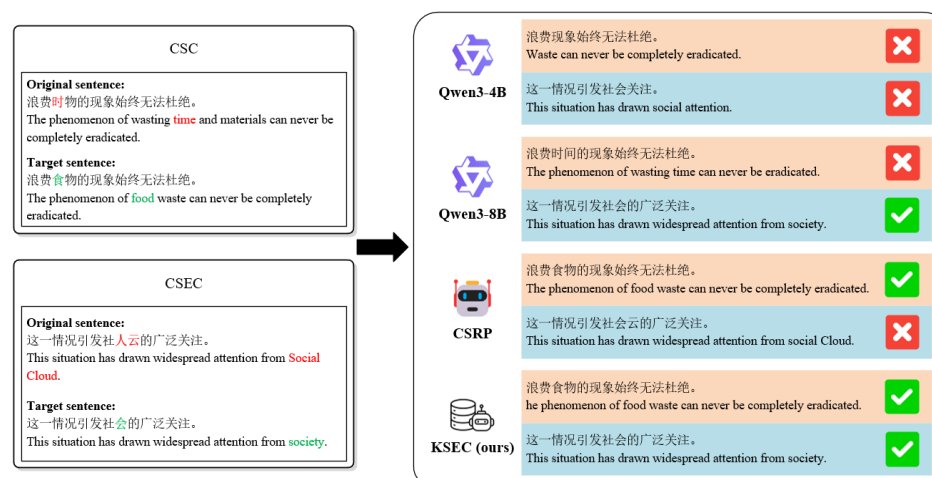


Figure 6. Examples of inference output from different models for the same cases.

In Figure 7, for the traditional CSC task, the attention matrix of KSEC shows a highly concentrated alignment of the erroneous positions. In contrast, the baseline LLMs show overly dispersed or biased attention weights, resulting in missed detections or overcorrection. Figure 8 shows the perception of different models for splitting errors. KSEC has well captured the remote dependencies of splitting characters, effectively focusing attention within the error range and guiding the merge operation. The baseline model failed to pay sufficient attention to the error area.

In summary, both the output comparisons and the visualized attention heatmaps confirm that KSEC can accurately attend to and resolve Chinese spelling errors as well as Chinese splitting errors, showing its ability to handle both conventional spelling errors and splitting errors.

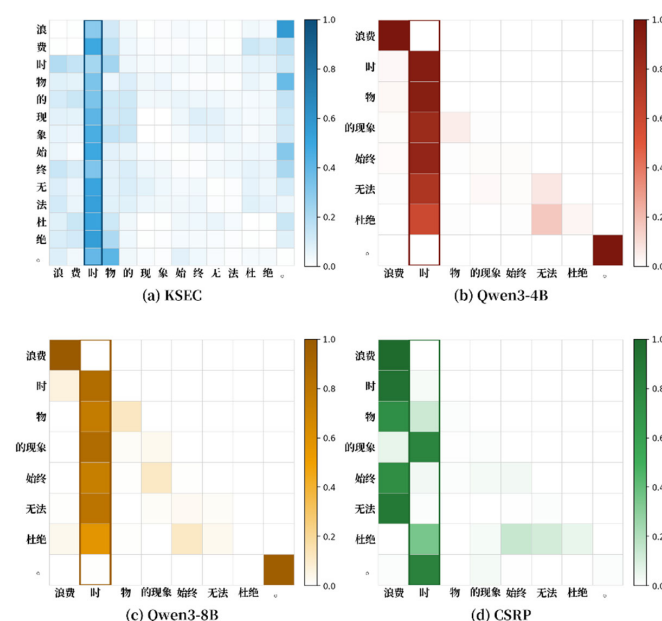


Figure 7. The visual heatmap of the attention distribution in the CSC case-based reasoning process. (a) Visualization heatmap of attention weights from KSEC; (b) visualization heatmap of attention weights from Qwen3-4B; (c) visualization heatmap of attention weights from Qwen3-8B; and (d) visualization heatmap of attention weights from CSRP.

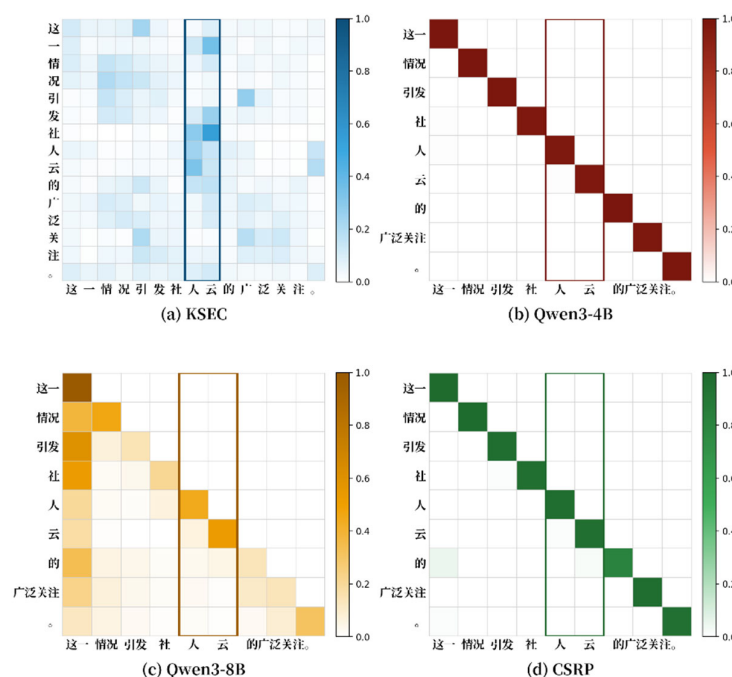


Figure 8. The visual heatmap of the attention distribution in the CSEC case-based reasoning process. (a) Visualization heatmap of attention weights from KSEC; (b) visualization heatmap of attention weights from Qwen3-4B; (c) visualization heatmap of attention weights from Qwen3-8B; and (d) visualization heatmap of attention weights from CSRP.

4. Discussion

The experimental results support our central hypothesis that a single encoder-only model can effectively handle both conventional same-length spelling errors and length-changing splitting errors. Crucially, this unified capability is achieved without compromising the baseline performance on standard CSC tasks.

These findings also highlight several promising directions for future work. First, the hybrid knowledge base, which currently covers component sequences of lengths two and three extracted from a fixed corpus, can be expanded. Integrating additional data sources and continuously updating the knowledge base with emerging online splitting patterns will further broaden its coverage. Since no standardized OCR or L2-learner benchmark is currently available for Chinese splitting-error correction, future work should also evaluate the model on naturally occurring splitting errors from real OCR outputs and learner corpora. Second, the source-aligned, deletion-based sequence-labeling formulation can be extended to other types of length-changing edits beyond character splitting. Finally, the proposed knowledge-injection mechanism can be paired with stronger pretrained backbones as more advanced models become available. These potential future extensions are viewed as pathways to broaden the reach of the framework.

To better interpret the CSEC results, we quantified the relation between the two resources. Among the 2469 SKB entries, 2036 entries (82.5%) share an identical mapping with the CSEC generation dictionary, while 433 entries are additional or differently mapped. This overlap largely reflects their shared use of Chinese character decomposition knowledge. Importantly, a dictionary match alone does not determine whether a span should be merged in context; the model must still identify the error, avoid false merges, and recover the exact sentence. Among the 158 false negatives on news, 103 retain all gold spans after pre-detection, 48 involve SKB coverage gaps, and only seven results from greedy conflicts. Since weak MLM candidates are retained, the MLM step causes no hard rejection. These

results show that pre-detection is generally reliable and that further gains mainly depend on improving final contextual prediction.

5. Conclusions

This paper introduces KSEC, a knowledge-enhanced framework that unifies Chinese spelling-error correction with Chinese splitting-error correction. To achieve this, we first construct a hybrid splitting knowledge base by combining a structural character table with corpus-mined confusion sets. Next, we reformulate length-changing correction as a source-aligned sequence-labeling task, utilizing an encoder-only backbone equipped with an explicit deletion token. Finally, we integrate the specialized knowledge into the model through a split-aware token-type bias and a tailored confusion loss function. Specifically, it effectively resolves over-splitting phenomena that are common in social media text and OCR outputs. Experiments show that KSEC provides robust splitting correction across formal and informal domains, with a particular advantage on noisy social text. It outperforms SoMu on social, while SoMu remains stronger on the cleaner news domain.

Author Contributions: Conceptualization, J.W. and G.H.; methodology, J.W.; software, J.W.; validation, J.W., Y.W., Q.G. and Y.L.; formal analysis, J.W. and Y.W.; investigation, J.W. and Q.G.; resources, G.H.; data curation, J.W. and Y.L.; writing—original draft preparation, J.W.; writing—review and editing, G.H. and J.W.; visualization, Y.W. and N.D.; supervision, G.H.; project administration, G.H.; funding acquisition, G.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Guangxi Qingmiao Talent Research Project, NO. C24RSN05SX03; the Guangxi Youth Science Foundation Project, NO. 2026GXNSFBA00640380; and the Guangxi Key Research and Development Plan Project, No. GuiKeAB24010327.

Data Availability Statement: The data presented in this study can be obtained from the following sources: CSC, <https://github.com/TW-NLP/ChineseErrorCorrector> (accessed on 20 May 2026), reference number [33]; CSEC, <https://github.com/Skywalker-Harrison/SoMu> (accessed on 16 March 2026), reference number [31].

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

KSEC	Knowledge-enhanced Splitting Error Corrector
CSC	Chinese Spelling Correction
CSEC	Chinese Splitting Error Correction
SKB	Splitting Knowledge Base
LLMs	Large Language Models

References

1. Martins, B.; Silva, M.J. Spelling correction for search engine queries. In *International Conference on Natural Language Processing*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 372–383. (In Spain)
2. Afli, H.; Qui, Z.; Way, A.; Sheridan, P. Using SMT for OCR error correction of historical texts. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation*, Portorož, Slovenia, 23–28 May 2016; pp. 962–966.
3. Hinton, G.; Deng, L.; Yu, D. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Process. Mag.* **2012**, *29*, 82–97. [[CrossRef](#)]
4. Zhang, S.; Huang, H.; Liu, J. Spelling error correction with soft-masked BERT. *arXiv* **2020**, arXiv:2005.07421.
5. Gomes, A.; Mendes, A.J.; Marcelino, M.J.; Teixeira, A. A teacher’s view about introductory programming teaching and learning—Portuguese and Macanese perspectives. In *Proceedings of the IEEE Frontiers in Education Conference*; IEEE: New York, NY, USA, 2017; pp. 1–8.

6. Liang, J.; Zhu, J.; Zhai, F.; Cheng, N.; Zong, C.; Zhou, Y. A hybrid approach towards Chinese spelling and splitting error correction. In Proceedings of the ECAI 2024: 27th European Conference on Artificial Intelligence, Santiago de Compostela, Spain, 19–24 October 2024; pp. 3883–3890.
7. Huang, W.; Dong, X.; Wang, M.; Liu, G.; Yu, J.; Zhu, H.; Yin, J. CSEC: A Chinese Semantic Error Correction Dataset for Written Correction. In *International Conference on Neural Information Processing*; Springer Nature: Singapore, 2023; pp. 383–398.
8. Li, J.; Wang, Q.; Mao, Z. Improving Chinese spelling check by character pronunciation prediction: The effects of adaptivity and granularity. *arXiv* **2022**, arXiv:2210.10996.
9. Lv, Q.; Cao, Z.; Geng, L. General and domain-adaptive Chinese spelling check with error-consistent pretraining. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* **2023**, *22*, 1–18. [[CrossRef](#)]
10. Devlin, J.; Chang, M.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MN, USA, 2–7 June 2019; Volume 1, pp. 4171–4186.
11. Li, X.; Du, H.; Zhao, Y. Towards Robust Chinese Spelling Check Systems: Multi-round Error Correction with Ensemble Enhancement. In *Proceedings of the CCF International Conference on Natural Language Processing and Chinese Computing*; Springer: Cham, Switzerland, 2023; pp. 325–336.
12. Sun, R.; Wu, X.; Wu, Y. An error-guided correction model for Chinese spelling error correction. *arXiv* **2023**, arXiv:2301.06323.
13. Omelanchuk, K.; Atrasevych, V.; Chernodub, A.; Skurzshanskyi, O. GECToR—Grammatical error correction: Tag, not rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2020; pp. 163–170.
14. Zhao, W.; Wang, L.; Shen, K.; Jia, R.; Liu, J. Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2019; pp. 156–165.
15. Xu, L.; Wu, J.; Peng, J.; Fu, J.; Cai, M. FCGEC: Fine-grained corpus for Chinese grammatical error correction. In *Findings of the Association for Computational Linguistics: EMNLP 2022*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2022; pp. 1900–1918.
16. Li, J.; Guo, J.; Zhu, Y.; Sheng, X.; Jiang, D.; Ren, B.; Xu, L. Sequence-to-action: Grammatical error correction with action guided sequence generation. In Proceedings of the AAAI Conference on Artificial Intelligence, Online, 22 February–1 March 2022; Volume 36, pp. 10974–10982.
17. Zhang, D.; Li, Y.; Zhou, Q. Contextual similarity is more valuable than character similarity: An empirical study for Chinese spell checking. In Proceedings of the ICASSP 2023—2023 IEEE International Conference on Acoustics, Speech and Signal Processing, Rhodes, Greece, 4–10 June 2023; pp. 1–5.
18. Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I. Gpt-4 technical report. *arXiv* **2023**, arXiv:2303.08774.
19. Wang, X.; Zhao, R.; Dai, H. An Empirical Investigation of Domain Adaptation Ability for Chinese Spelling Check Models. In Proceedings of the ICASSP 2024—2024 IEEE International Conference on Acoustics, Speech and Signal Processing, Seoul, Republic of Korea, 14–19 April 2024; pp. 9996–10000.
20. Wang, F.; Shao, H.; Xie, Z. AxBERT: An explainable Chinese spelling correction method driven by associative knowledge network. In Proceedings of the ICLR, Kigali, Rwanda, 1–5 May 2023.
21. Liu, L.; Wu, H.; Zhao, H. Chinese Spelling Correction as Rephrasing Language Model. *Proc. AAAI Conf. Artif. Intell.* **2024**, *38*, 18662–18670. [[CrossRef](#)]
22. Yu, L.C.; Lee, L.H.; Tseng, Y.H.; Chen, H.H. Overview of SIGHAN 2014 bake-off for Chinese spelling check. In Proceedings of the Third CIPS-SIGHAN Joint Conference on Chinese Language Processing, Wuhan, China, 20–21 October 2014; pp. 126–132.
23. Tseng, Y.H.; Lee, L.H.; Chang, L.P.; Chen, H.H. Introduction to SIGHAN 2015 bake-off for Chinese spelling check. In Proceedings of the Eighth SIGHAN Workshop on Chinese Language Processing, Beijing, China, 30–31 July 2015; pp. 32–37.
24. Hu, Y.; Meng, F.; Zhou, J. Cscd-ime: Correcting spelling errors generated by pinyin ime. *arXiv* **2022**, arXiv:2211.08788.
25. Wang, D.; Song, Y.; Li, J.; Han, J.; Zhang, H. A hybrid approach to automatic corpus generation for Chinese spelling check. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018; pp. 2517–2527.
26. Xu, H.D.; Li, Z.; Zhou, Q.; Li, C.; Wang, Z.; Cao, Y.; Huang, H.; Mao, X.L. Read, listen, and see: Leveraging multimodal information helps Chinese spell checking. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2021; pp. 716–728.
27. Zhang, Y.; Li, Z.; Bao, Z.; Li, J.; Zhang, B.; Li, C.; Huang, F.; Zhang, M. MuCGEC: A multireference multi-source evaluation dataset for Chinese grammatical error correction. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Seattle, WA, USA, 10–15 July 2022; pp. 3118–3130.

28. Liu, S.; Yang, T.; Yue, T. PLOME: Pre-training with misspelled knowledge for Chinese spelling correction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2021; pp. 2991–3000.
29. Xu, H.; Li, Z.; Zhou, Q. Read, listen, and see: Leveraging multimodal information helps Chinese spell checking. *arXiv* **2021**, arXiv:2105.12306.
30. Cheng, X.; Xu, W.; Chen, K. Spellgcn: Incorporating phonological and visual similarities into language models for Chinese spelling check. *arXiv* **2020**, arXiv:2004.14166.
31. Yang, A.; Xiao, B.; Wang, B.; Zhang, B.; Bian, C.; Yin, C.; Lv, C.; Pan, D.; Wang, D.; Yan, D.; et al. Baichuan 2: Open large-scale language models. *arXiv* **2023**, arXiv:2309.10305.
32. Yang, A.; Li, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Gao, C.; Huang, C.; Lv, C.; et al. Qwen3 technical report. *arXiv* **2025**, arXiv:2505.09388.
33. Tian, W.; Zhou, Y.; Lan, M. CSRP: Chain-of-Thought Reasoning for Chinese Text Correction via Reinforcement Learning with Efficiency-Aware Rewards. *arXiv* **2026**, arXiv:2606.00020.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.