Chinese Spelling Check with Nearest Neighbors

Anonymous ACL submission

Abstract

Chinese Spelling Check (CSC) aims to detect and correct error tokens in Chinese contexts, which has a wide range of applications. In this paper, we introduce **InfoKNN-CSC**, which extends the standard **CSC** model by linearly interpolating it with a k-nearest neighbors (k**NN**) model. Moreover, the phonetic, graphic, and contextual information (**info**) of tokens and contexts are elaborately incorporated into the design of the query and key of kNN, according to the characteristics of the task. After retrieval, in order to match the candidates more accurately, we also perform reranking methods based on the overlap of the n-gram values and inputs. Experiments on the SIGHAN benchmarks demonstrate that the proposed model achieves state-of-the-art performance with substantial improvements over existing work.

1 Introduction

The purpose of Chinese spelling check (CSC) is to detect and correct spelling errors in Chinese text, which often occur between characters with similar phonetics and morphology. The research on CSC is significant since it benefits various NLP tasks, such as speech recognition, optical character recognition, data cleaning, Chinese grammar error correction, and so on. With the development of deep learning and pretrained language models, great progress has been made in this task (Etoori et al., 2018; Guo et al., 2019; Zhang et al., 2020). Further, many current works have turned to introducing phonological and visual information into their models (Nguyen et al., 2020; Wang et al., 2021; Zhang et al., 2021). Their methods are based on statistics from Liu et al. (2011) that 83% of Chinese spelling errors are caused by phonological similarity, 48% are due to visual similarity and 35% involve both factors.

However, Chinese spelling check is still challenging because it suffers from subtle and diverse errors. Furthermore, we speculate that current methods have not fully utilized the training data, let alone the lack of an adequate parallel corpus. As shown in Table 1, the existing model REALISE (Xu et al., 2021) fails to correct "设(set)" to "这(this)" at test time though the same error ("设" → "这") in similar contexts occurs a few times in the training set. Meanwhile, the model cannot correct the token "偷(steal)" while its correct usage also appears. To make better use of the information in the dataset, we introduce the retrieval-augmented method with an elaborately designed k-nearest neighbors (kNN) model and reranking mechanism.

Error Pair in Training Set	Input	因为设(set)是校长的工作。
	Correct	因为这(this)是校长的工作。
	Model Output	因为涉(relate)是校长的工作。
	Translation	Because this is the principal's job.
	Samples in Traing Set	设(这)是她主演的电影. 我们设(这)个周末见面. ...
Correct Usage in Training Set	Input	旁边的人头(head)了我的手册。
	Correct	旁边的人偷(steal)了我的手册。
	Model Output	旁边的人投(cast)了我的手册。
	Translation	The person next to me stole my manual.
	Samples in Traing Set	有人偷了我的钱包。 有个女生偷了我的东西。 ...

Table 1: Examples of Chinese spelling errors, including inputs, target outputs(correct), model outputs, and hints in the training set. The model shown here is REALISE (Xu et al., 2021), one of the strong baselines.

Retrieval-augmented text generation, a new generation paradigm known as "open-book exam", can be targeted to solve such problems by integrating deep learning models with traditional retrieval technologies (Guu et al., 2020; Weston et al., 2018; Gu et al., 2018). Among them, algorithms based on kNN retrieval always predict tokens with a nearest neighbor classifier over a large datastore of cached examples, using representations from a neural model for similarity search (Khandelwal et al., 2019, 2020; Kassner and Schütze, 2020). The kNN

retrieval algorithms for model improvement have proven effective for many tasks, such as machine translation, language modeling, dialogue generation, and so on. However, CSC has some significant differences compared with the above tasks, on the basis of which we propose our corresponding methods.

Above all, both correct and incorrect tokens exist in the input text, which makes it confusing and unreasonable to arbitrarily store the hidden representations of each token from the neural model for retrieval. As mentioned before, the incorrect token is often caused by phonological and visual similarity. So we incorporate the phonological and visual information of each token itself into the calculation of the key. Furthermore, the contextual information around the target token is also encoded according to a certain distribution. We suppose that phonological and visual information, fused with the contextual encoding, which we call *error-robust information* (denoted as *ERInfo*), is more robust and not as sensitive as the pure semantic information during retrieval. In addition, there are many overlaps between each pair of input and output texts in CSC since only a few tokens are incorrect. So we can store the n-gram around the target token as the value to construct the datastore for further filtering and reranking instead of conventionally just storing the token itself.

Combining the above two ideas, we retrieve the n-gram neighbors of the target token by its *error-robust information*, rerank them based on their overlap with the corresponding input n-gram, and finally get the word distribution over the vocabulary, which is called **InfoKNN** for convenience. We introduce **InfoKNN-CSC** which extends a pre-trained CSC model by linearly interpolating the original word distribution with the InfoKNN. The experimental results of InfoKNN-CSC on three SIGHAN benchmarks surpass those of the previous methods. Furthermore, thanks to the design of *ERInfo*, we can expand the data more easily by adding non-parallel texts to the datastore directly than other methods that need to construct pseudo-data with confusion sets.

In summary, our contributions are as followed:

1) To our best knowledge, our work is the first to employ the retrieval-augmented method on Chinese spelling check task, which can be used in a plug-and-play manner without training and allows more flexible expansion of the datastore.

2) We elaborately design the specific key and value in the datastore and propose InfoKNN to fuse richer information for more robust retrieval.

3) The experiment shows that our model achieves state-of-the-art performance on the SIGHAN datasets with substantial improvement. The code will be released to the community.

2 Background

2.1 Nearest Neighbor Language Modeling

Given a context sequence $c_t = (w_1, \cdots, w_{t-1})$, the language model estimates the distribution over the target token $p_{\mathrm{LM}}(w_t \mid c_t)$. Khandelwal et al. (2019) proposed kNN-LM to involve augmenting the pre-trained LM with a nearest neighbors retrieval mechanism.

Firstly, let $f(\cdot)$ be the function that maps a context c to a fixed-length vector representation. Therefore, we can use the training set $\mathcal{D}$ to build the datastore:

$$(\mathcal{K}, \mathcal{V}) = \{(f(c_i), w_i) \mid (c_i, w_i) \in \mathcal{D}\} \quad (1)$$

Then at step t during inference, given the input context c_t, the model queries the datastore with $f(c_t)$ to retrieve its k-nearest neighbors $\mathcal{N}$ using a distance function $d(\cdot, \cdot)$ and then gets the target token's probability over the vocabulary:

$$p_{\mathrm{kNN}}(w_{t+1} \mid c_t) \propto \sum_{(k_i, v_i) \in \mathcal{N}} \mathbb{I}_{w_{t+1}=v_i} \exp\left(-d\left(k_i, f(c_t)\right)\right). \quad (2)$$

Finally, the distribution obtained by kNN will be interpolated to the standard LM distribution:

$$p(w_{t+1} \mid c_t) = \lambda p_{\mathrm{kNN}}(w_{t+1} \mid c_t) + (1-\lambda) p_{\mathrm{LM}}(w_{t+1} \mid c_t). \quad (3)$$

2.2 Chinese Spelling Check

The goal of the standard CSC model is to learn the conditional probability $p_{\mathbf{csc}}(\mathbf{y} \mid \mathbf{x})$ for correcting a sentence $\mathbf{x} = \{x_1, \cdots, x_n\}$ which includes errors to a corresponding correct sentence $\mathbf{y} = \{y_1, \cdots, y_n\}$. Correction is typically performed in a masked language modeling manner, and the probability of each predicted token can be factored as $p_{\mathbf{csc}}(y_i \mid \mathbf{x})$.

The CSC model always encodes $\mathbf{x}$ into the hidden states $\mathbf{h}$. Due to the characteristics of CSC, error correction usually requires phonetic and graphic

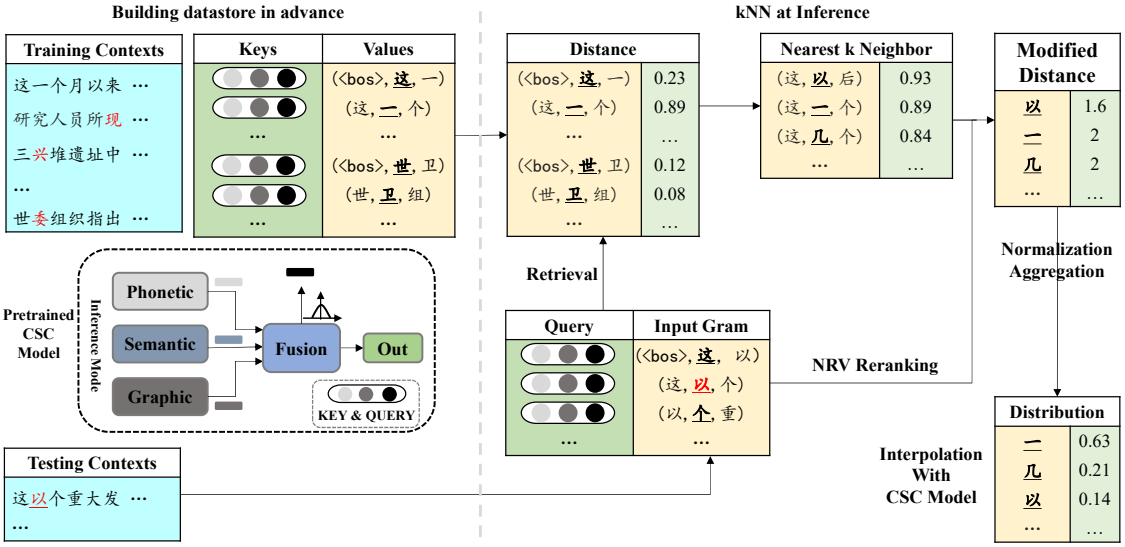

Figure 1: An illustration of InfoKNN-CSC. We use all the tokens in the training set to make the datastore. The key includes the phonetic, graphic and contextual information of the token obtained from the pre-trained CSC model, and the value is in the form of an n-gram. There are both correct tokens (the majority) and incorrect tokens (marked in red) in the training data, such as "现:now" (which should be changed to "观:observe") and "委:committee" (which should be changed to "卫:health"). The test sample in the figure shows the correction process for the token "以(to)" in "这以个(this to)", which should be changed to "一(one)".

information for each token. So the CSC model usually also obtains hidden representations of font images and pronunciations of input tokens, which are denoted as $\mathbf{p}$ and $\mathbf{g}$.

Token representations are obtained after fusing these hidden representations using an appropriate approach, such as a gating mechanism, denoted as $\mathbf{m}$:

$$\mathbf{m} = G(\mathbf{h}, \mathbf{p}, \mathbf{g}). \tag{4}$$

CSC model will output probability distribution of each token over vocabulary $\mathcal{V}$ according to the corresponding hidden representation,

$$p_{\text{csc}}\left(y_i \mid x_i\right) = O(m_i), \tag{5}$$

where O is a predict function, such as a linear transform followed by Softmax.

Since Chinese Spelling Check can be performed in a masked language modeling manner, similar to language modeling, we can enhance the CSC model with a nearest neighbor retrieval mechanism. Based on this idea, it is natural to make a datastore using the hidden vectors in the CSC model as dense indexes and then perform retrieval to integrate with the CSC model. However, due to the characteristics of the CSC task, such as the mix of correct and incorrect tokens in the input, we need to make some adjustments with full consideration, which will be described in Section 3.

3 Methodology

As shown in Figure 1, the core idea of our work is to enhance the CSC model with a nearest neighbor retrieval mechanism. The datastore used for retrieval is carefully designed according to the characteristics of CSC task to enhance the robustness of retrieval and make better use of every token, no matter it is correct or incorrect. In Section 3.1, we introduce the method of building the datastore. In Section 3.2, we introduce our method to utilize kNN search results.

3.1 Datastore Building

The structure of the datastore is a dictionary, in which each element consists of the pair *(key, value)*. The key is used to retrieve the nearest neighbor and obtain the value which contains the information we need.

Key Design The goal of our key design is to avoid mixing correct and incorrect tokens in the input and provide sufficient information for error correction. Each token needs to be represented more rationally and robustly in the same high-dimensional space.

We take the last sentence "世委组织指出...(The World Commission states that...)" in the training context in Figure 1 as an example. It should be cor-

rected to "世卫组织指出...(WHO states that...)", where "委" is wrong and has a similar pronunciation ("wei") to "卫". We believe that using the semantic information of the wrong token is misleading and unreliable in the process of retrieval, but its phonetic or morphological information is usually approximately correct, according to the study of Liu et al. (2011). Therefore, it is more robust to use the information about phonology and morphology.

Furthermore, to be able to correct errors, it is not enough to only use phonetic and graphic information, but contextual information should also be added. In this example, what is helpful for the correction is that the context of "委" is "世(world)" and "组织(organization)" and the pronunciation of "委" is "wei". Based on the information above, we can infer that it should be "世卫组织(WHO)". Although the word representation in the semantic encoder has integrated contextual information to some extent, we suppose it is not robust enough because it is mainly influenced by the input word, which may be wrong. Therefore, we would like to use a more careful approach to obtain contextual information. Here, we use a Gaussian distribution to weight the hidden representations of tokens around the current token to obtain the contextual representation.

Formally, given an incorrect-correct sentence pair $(x, y) \in (\mathcal{X}, \mathcal{Y})$ in the training set, a CSC model corrects the t-th input token x_t to the t-th target token y_t based on the input context x. we denote the hidden representation of the input token x_t's hidden representation of its pronunciation, morphology and semantics as $ph(x_t)$, $mo(x_t)$ and $s(x_t)$, respectively. They can be obtained separately with the phonetic, graphic, and semantic encoders, and they are fused as follows:

$$f(x_t) = f(ph(x_t), mo(x_t), s(x_t)), \quad (6)$$

where the fusion function f is usually a gate mechanism. The contextual representation of x_t we designed is obtained by weighting its neighboring words' representations:

$$c(x_t) = \sum_{0 \le i \le L} f_n(i; t, \sigma) f(x_i), \quad (7)$$

$$f_n(i; t, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(i-t)^2}{2\sigma^2}), \quad (8)$$

where L is the length of input sequence x, f_n is the probability density function of Gaussian distribution.

In the end, we use concatenation to combine and store three parts of information and obtain the key that satisfies our needs,

$$k_t = [ph(x_t); mo(x_t); c(x_t)], \quad (9)$$

We call the information stored in the key *error-robust information* (denoted as *ERInfo*). For convenience, the design of the key is called **InfoK**.

Value Design The standard kNN-LM is designed to only use the target token as the value, which is relatively simple. Considering the test sample "这以个(this to)" in Figure 1, we can find that using the *ERInfo* to retrieve is still not enough. It always gets lots of same target tokens as the query token. It may be because they have the same phonology and morphology representation in *ERInfo*.

Furthermore, we find that we can rely on the context of the the input token for further filtering. In other words, we believe that the retrieved neighbors with the same context as the input token should be more accurate and useful. For example, "这一个(this one)" is more likely to be the target than "这以后(after this)" for "这以个(this to)".

Therefore, we decide to store the n-gram with a window of size n centered on the corresponding token y_t in the target output to store contextual information more explicitly. Formally, we obtain the value,

$$v_t = [y_{t-\lfloor n/2 \rfloor}, \cdots y_t, \cdots, y_{t+\lfloor n/2 \rfloor}]. \quad (10)$$

For convenience, we call the design that stores **n**-gram **v**alues as contextual information and then uses them for further **r**eranking **NRV**.

In this way, the datastore is built from the training set,

$$(\mathcal{K}, \mathcal{V}) = \bigcup_{(x,y) \in (\mathcal{X}, \mathcal{Y})} \{(k_t, v_t), \forall y_t \in y\}. \quad (11)$$

3.2 KNN Retrieval and Utilization

Retrieval During inference, for each token x_t, the InfoKNN-CSC model aims to predict $\hat{y}_t$, given the input sequence x as well as x_t's hidden representation $f(x_t)$ and the *ERInfo* representation $ph(x_t)$, $mo(x_t)$ and $c(x_t)$. We use these representations to generate the query q_t in the same way in Section 3.1,

$$q_t = [ph(x_t); mo(x_t); c(x_t)],$$

which will be used to retrieve the k nearest neighbors in the datastore with respect to the l_2 distance.

After retrieval we can get k nearest neighbors $N_t = \{(k_i, v_i), i \in \{1, 2, \cdots k\}\}$ and the distance $D_t = \{d(q_t, k_i), i \in \{1, 2, \cdots k\}\}$, where $d(q_t, k_i)$ means l_2 distance between k_i and q_t.

Reranking Remember that the value in the datastore is in the form of n-gram, so we can use the nature of the CSC where the input and output overlap a lot to rerank the k retrieved nearest neighbors.

For each input token x_t, we can obtain its n-gram $g_t = [x_{t-\lfloor n/2 \rfloor}, \cdots x_t, \cdots, x_{t+\lfloor n/2 \rfloor}]$ with a window of size n centered on x_t. For the x_t's j-th neighbor $N_t(j)$, we denote the distance between it and x_t as $D_t(j)$. We modify the distance $D_t(j)$ based on the n-gram overlap between the input x_t and the neighbor $N_t(j)$'s value v_j:

$$\begin{aligned} \alpha_t &= \frac{\sum_{1 \le i \le n} \mathbb{I}(v_j^i, g_t^i) w^i}{n}, \\ D'_t(j) &= (1 - \alpha_t) D_t(j), \end{aligned} \tag{12}$$

where w is the gain of each position if they are same, and α_t represents how much the retrieved n-gram overlaps with the input, which can measure their similarity.

Utilization With the above designs, the target word's probability distribution over the vocabulary based on the retrieved neighbors is computed as:

$$\begin{aligned} &p_{\text{kNN}}(y_t \mid x_t) \propto \\ &\sum_{(k_i, v_i)} \mathbb{I}(y_t = v_i^{\lfloor k/2 \rfloor}) \exp\left(\frac{-D'_t(i)}{T}\right), \end{aligned} \tag{13}$$

where T is the softmax temperature and $v_i^{\lfloor k/2 \rfloor}$ is the central word of v_i. The final probability when predicting y_t is calculated as the interpolation of two distributions with a hyper-parameter λ:

$$\begin{aligned} p(y_t \mid x_t) = &\lambda p_{\text{kNN}}(y_t \mid x_t) \\ &+ (1 - \lambda) p_{\text{csc}}(y_t \mid x_t) \end{aligned} \tag{14}$$

where p_{csc} indicates the vanilla CSC model's prediction.

4 Experiment

In this section, we introduce the details of experiments, including datasets, metrics, baselines, and the main results we obtained. Then we conduct analysis and discussions to verify the effectiveness of our method.

4.1 Datasets

Training Data We use the same training data by following previous works (Zhang et al., 2020; Liu et al., 2021; Xu et al., 2021; Li et al., 2022c), including the training samples from SIGHAN13 (Wu et al., 2013), SIGHAN14 (Yu et al., 2014), SIGHAN15 (Tseng et al., 2015) and the pseudo training data, denoted as Wang271K (Wang et al., 2018).

In addition, we randomly select 10% of the training data during training as our verification set to select the best hyperparameters.

Test Data To guarantee fairness, we use the same test data as previous work, which are from the SIGHAN13/14/15 test datasets. It is noted that the text of the original SIGHAN dataset is in Traditional Chinese, so we use OpenCC to preprocess these original datasets into Simplified Chinese which has been widely used in previous work (Wang et al., 2019; Cheng et al., 2020; Zhang et al., 2020; Xu et al., 2021). Detailed statistics of the training/test data we used in our experiments are shown in Appendix A.

4.2 Evaluation Methods

We evaluate our model's predictions with the sentence-level metrics which was used in most of the previous work. The results are reported at both detection level and correction level. At the detection level, a sentence is considered correct if all spelling errors in the sentence are successfully detected. At the correction level, the spelling errors not only need to be detected, but also need to be corrected. We report accuracy, precision, recall, and F1 scores at both levels. To facilitate comparisons in the later works, we also report our results using the official SIGHAN tool and results in character-level metrics in Appendix C.

4.3 Baseline Models

To evaluate the performance of InfoKNN-CSC, we select several advanced strong baseline methods: **FASpell** designed by Hong et al. (2019) is a model that consists of a denoising autoencoder and a decoder. **SpellGCN** (Cheng et al., 2020) integrates the confusion set to the correction model through GCNs to improve CSC performance. **PLOME** (Liu et al., 2021) is a task-specific pretrained language model to correct spelling errors. **REALISE** (Xu et al., 2021) is a multimodel CSC model which captures and mixes the semantic, phonetic and

Dataset	Model	Detection Level				Correction Level			
		Acc	**Pre**	**Rec**	**F1**	**Acc**	**Pre**	**Rec**	**F1**
SIGHAN13	FASpell (Hong et al., 2019)	63.1	76.2	63.2	69.1	60.5	73.1	60.5	66.2
	SpellGCN (Cheng et al., 2020)	-	80.1	74.4	77.2	-	78.3	72.7	75.4
	ECOPO† (Li et al., 2022c)	83.3	89.3	**83.2**	86.2	82.1	88.5	82.0	85.1
	REALISE† (Xu et al., 2021)	82.7	88.6	82.5	85.4	81.4	87.2	81.2	84.1
	InfoKNN-CSC† (Ours)	**83.4**	**90.0**	82.8	**86.3**	**82.3**	**89.1**	**82.2**	**85.6**
SIGHAN14	FASpell (Hong et al., 2019)	70.0	61.0	53.5	57.0	69.3	59.4	52.0	55.4
	SpellGCN (Cheng et al., 2020)	-	65.1	69.5	67.2	-	63.1	67.2	65.3
	ECOPO (Li et al., 2022c)	79.0	68.8	**72.1**	70.4	78.5	67.5	**71.0**	69.2
	REALISE (Xu et al., 2021)	78.4	67.8	71.5	69.6	77.7	66.3	70.0	68.1
	InfoKNN-CSC (Ours)	**79.9**	**72.1**	70.6	**71.3**	**79.6**	**71.3**	69.8	**70.6**
SIGHAN15	FASpell (Hong et al., 2019)	74.2	67.6	60.0	63.5	73.7	66.6	59.1	62.6
	SpellGCN (Cheng et al., 2020)	-	74.8	80.7	77.7	-	72.1	77.7	75.9
	PLOME (Liu et al., 2021)	-	77.4	81.5	79.4	-	75.3	79.3	77.2
	ECOPO (Li et al., 2022c)	85.0	77.5	**82.6**	80.0	84.2	76.1	**81.2**	78.5
	SCOPE‡ (Li et al., 2022b)	-	80.2	83.2	**81.7**	-	77.5	80.4	78.9
	REALISE (Xu et al., 2021)	84.7	77.3	81.3	79.3	84.0	75.9	79.9	77.8
	InfoKNN-CSC (Ours)	**86.1**	**81.1**	81.3	81.2	**85.6**	**79.9**	80.1	**80.0**

Table 2: Sentence-level performance of InfoKNN-CSC and all baseline methods. REALISE is the backbone for InfoKNN-CSC to build the datastore. Results marked with "†" on SIGHAN 2013 are post-processed with removing all "的", "地", "得" from the model output, due to the low annotation quality about them which is to follow the previous work (Xu et al., 2021) for convenient comparison. Results marked with "‡" are obtained by using the same data as our model to implement SCOPE. The detailed comparison with SCOPE is shown in Appendix D.

graphic information. **ECOPO** (Li et al., 2022c) is an error-driven contrastive probability optimization framework and can be combined with other CSC models. **SCOPE** (Li et al., 2022b), concurrently to our work, consists of a shared encoder and two parallel decoders that introduces an auxiliary task of Chinese pronunciation prediction. Note that SCOPE uses additional training data (wiki2019zh[1]) compared to the other work. For fair comparison, we show the results of our model using the same data as SCOPE in the Appendix D.

4.4 Implementation Details

To get the *ERInfo* that is required to construct our datastore, we consider using a pre-trained model on the current task, like other kNN retrieval-related work does. We choose REALISE, a multimodel model that captures and mixes semantic, phonetic, and graphic information, which meets our requirements very well.

More specifically, a pre-trained GRU encodes the pinyin sequence of input text to obtain the phonetic information $ph(x_t)$ of each token, and a pre-trained ResNet encodes the character image to obtain the corresponding graphic information $mo(x_t)$. Considering the trade-off between storage space and model performance, we store 3-gram of every central token as the value of datastore. We implement the grid search on the validation set to determine the hyperparameters of our experiments, and more details are shown in Appendix B.

4.5 Experimental Results

The results on sentence-level metrics of InfoKNN-CSC and all baseline methods are shown in Table 2. We also report our results using the official SIGHAN tool and results on character-level metrics in Appendix C. We can observe that the InfoKNN-CSC has obtained substantial improvements on SIGHAN14 and SIGHAN15 while achieving comparable results on SIGHAN13, compared to the previous state-of-the-art model ECOPO. When turning to the REALISE, on which our model is based, the improvement is more remarkable, with about a

[1] https://github.com/brightmart/nlp_chinese_corpus

SIGHAN15	Detection-level			Correction-level		
	D-P	D-R	D-F	C-P	C-R	C-F
InfoKNN	81.1	81.3	81.2	79.9	80.1	80.0
w/o Info-P	79.3	80.9	80.1	78.3	79.4	78.9
w/o Info-G	79.5	81.0	80.2	78.7	79.5	79.1
w/o Info-C	80.4	81.1	80.7	79.2	80.2	79.7
w/o InfoK	77.7	81.3	79.5	76.5	80.0	78.2
w/o NRV	79.7	81.2	80.4	78.4	79.9	79.1
w/o kNN	77.3	81.3	79.3	75.9	79.9	77.8

Table 3: Ablation results of the InfoKNN-CSC model on SIGHAN2015 test set. We apply the following changes to InfoKNN-CSC: 1) removing each element of *ERInfo*(w/o Info-P, w/o Info-G, w/o Info-C denote the reduction of phonetic, graphic and contextual information respectively.); 2) using the hidden representation of the token as the key (w/o InfoK); 3) only using the target token as the value(w/o NRV); 4) removing the kNN module(w/o kNN)

2.0% average increase on three SIGHAN datasets.

On the other hand, it is notable that both the accuracy and the precision of our model have improved remarkably, while the recall score has no great change. It demonstrates that the model becomes less prone to wrong corrections which may be due to the fact that the model remembers more correct samples. More detailed analysis is provided in the Section 5.

5 Analysis and Discussion

5.1 Ablation Experiments

We conduct ablation experiments to analyze the effects of the components of *ERInfo* and the design of InfoK and NRV on the performance of our method. The results are shown in Table 3.

We can see that all three types of information in *ERInfo* are critical, especially the phonetic information. It may be due to the fact that most of the errors in the sighan test set are phonological similarity errors. And there is less decrease when contextual information is removed, probably because the design of NRV also introduces contextual information into the model.

When the InfoK and NRV are removed, the performance drops significantly in both detection and correction level. Especially in error correction, the decrease in model effectiveness is more noticeable. It may be because the absence of reranking by the n-gram value can make the model unable to modify or keep tokens in the sentence confidently. Besides, missing the key that incorporates the *ERInfo* and using only the hidden representation of each token, the model will be confused about whether the input token is reliable.

5.2 Data Augmentation

We also experiment with adding additional correct sentences to the datastore and find that the results are further improved. It is worth pointing out that other CSC researches need to perform data augmentation by adding noise to the raw texts using the confusion set obtained with the rule-based approach. The quality of the pseudo-data cannot be guaranteed in this way, so it is likely to affect the performance of the model. And it requires retraining the model, which also consumes time and resources. Compared to this data augmentation method, our method is simpler and can add the correct text to the datastore directly without retraining. The details and results are shown in Table 9 in Appendix D.

5.3 Effect of Key Hyperparameters

While InfoKNN requires no additional training, there are some hyperparameters still introduced. As shown in Appendix E, we investigate how key hyperparameters affect model's performance.

Number of Neighbors per Query As shown in Figure 2, the performance increases with the number of neighbors at first, and it starts to decrease when the number of neighbors increases to about 16. It may be because more noise will be introduced if too many neighbors are retrieved.

Softmax Temperature As shown in Figure 2, the performance is relatively robust to temperature and achieves good results over a wide range.

Interpolation Parameter As shown in Figure 3 in Appendix E, the model works best at $\lambda \approx 0.4$, probably because it can integrate the predictions of kNN and CSC model better.

5.4 Inference Time

As shown in Appendix F, we investigate the effect of InfoKNN on the inference time of CSC. We can see that InfoKNN causes the inference to be slightly slower, but the effect is not significant, probably because CSC model decodes in parallel.

5.5 Case Study

It can be seen that the presence of similar contexts in the training set causes the model to prefer to keep

Input:	老师进教师来了。
Correct:	老师就进教室来了。
Translation:	The teacher came into the classroom。
CSC Output:	老师就请教室来了。
InfoKNN Output:	老师就进教室来了。
Traing Sample:	当老师的第一个脚步踏进教室时...
Input:	我带上运动鞋出门。
Correct:	我带上运动鞋出门。
Translation:	I take my sneakers and go out。
CSC Output:	我戴上运动鞋出门。
InfoKNN Output:	我带上运动鞋出门。
Traing Sample:	...带上半亿珠宝现身北京。 ...被老师带上街头。

Table 4: Some examples from SIGHAN 2015. The word in red means an error, and the word in green means correct. "CSC Output" means the prediction from standard REALISE model.

the current token and therefore avoid incorrectly modifying it. That's why in Table 2 the precision score of the model has increased a lot.

As shown in Table 4, given an input, "老师就进教师来了", which means "The teacher entered the teacher", the standard CSC model REALISE not only changes "师(teacher)" to "室(room)" but incorrectly changes "进(entered)" to "请(invite)". Meanwhile the model argumented by the kNN avoids incorrect modifications successfully, benefit from a number of similar usages of "进" in the training set, such as "当老师的第一个脚步踏进教室时" which means "When the teacher's first footsteps entered the classroom".

Another example is "我带上运动鞋出门(.I take my sneakers and go out.)", which is correct, but REALISE incorrectly changed the "带(take)" in it to "戴(wear)" which is usually used in Chinese to refer to putting on a hat, glasses, etc. And InfoKNN-CSC do not make this mistake, because there are many similar uses of "带(take)" in the training set.

More similar examples can be found by comparing outputs of REALISE with our model.

6 Related Work

6.1 Chinese Spelling Check

CSC has received wide attention over the past decades. Early work (Mangu and Brill, 1997; Jiang et al., 2012) used manually designed rules to correct the errors. After that, methods based on statistical language models also made some progress (Yu and Li, 2014). With the development of deep learning and pretrained language model has achieved great improvements in recent years. FASpell (Hong et al., 2019) applied BERT as a denoising autoencoder for CSC. Soft-Masked BERT (Zhang et al., 2020) chose to combine a Bi-GRU based detection network and a BERT based correction network.

In recent times, many studies have attempted to introduce phonetic and graphic information into CSC models. SpellGCN was proposed to employ graph convolutional network on pronunciation and shape similarity graphs. Nguyen et al. (2020) employed TreeLSTM to get hierarchical character embeddings as graphic information. REALISE (Xu et al., 2021) used Transformer (Vaswani et al., 2017) and ResNet5 (He et al., 2016) to capture phonetic and graphic information separately. In this respect, PLOME (Liu et al., 2021) chose to apply the GRU (Bahdanau et al., 2014) to encode pinyin and strokes sequence. PHMOSpell (Huang et al., 2021) derived phonetic and graphic information from multi-modal pre-trained models including Tacotron2 and VGG19.

6.2 Retrieval-Augmented Paradigm

Retrieval-augmented text generation have been applied to many tasks including language modeling (Guu et al., 2020), dialogue (Weston et al., 2018), machine translation (Gu et al., 2018) and others. Li et al. (2022a) provide an overview of this paradigm.

Of these retrieval-augmented methods, the studies that most relevant to our paper are kNN-LM (Khandelwal et al., 2019), which extends a pre-trained neural language model by linearly interpolating it with a k-nearest neighbors model, kNN-NMT (Khandelwal et al., 2020), which combines k nearest neighbors algorithm closely with NMT models to improve performance, BERT-kNN (Kassner and Schütze, 2020) that interpolates BERT's prediction for question q with a kNN-search.

7 Conclusion

We propose the InfoKNN-CSC to improve the current CSC model with a nearest neighbor retrieval mechanism. The keys and values in the datastore for retrieval are carefully designed according to the characteristics of CSC to effectively make use of the data. The experimental results prove the effectiveness of our method and its improvement over previous work. Furthermore, because our method is simple to implement and can be combined with other CSC models without retraining, the performance can be continuously improved.

Ethics Statement

Chinese Spelling check usually won't present any ethical violation. Nevertheless, our datastore can be built from any open source corpus, and thus our model may still have a low risk of producing toxic content if there exists toxic text in the source text. Other limitations, such as disk space usage, are shown in Appendix G.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Xingyi Cheng, Weidi Xu, Kunlong Chen, Shaohua Jiang, Feng Wang, Taifeng Wang, Wei Chu, and Yuan Qi. 2020. Spellgcn: Incorporating phonological and visual similarities into language models for chinese spelling check. *arXiv preprint arXiv:2004.14166*.

Pravallika Etoori, Manoj Chinnakotla, and Radhika Mamidi. 2018. Automatic spelling correction for resource-scarce languages using deep learning. In *Proceedings of ACL 2018, Student Research Workshop*, pages 146–152.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor OK Li. 2018. Search engine guided neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Jinxi Guo, Tara N Sainath, and Ron J Weiss. 2019. A spelling correction model for end-to-end speech recognition. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5651–5655. IEEE.

Zhao Guo, Yuan Ni, Keqiang Wang, Wei Zhu, and Guotong Xie. 2021. Global attention decoder for chinese spelling error correction. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1419–1428.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International Conference on Machine Learning*, pages 3929–3938. PMLR.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Yuzhong Hong, Xianguo Yu, Neng He, Nan Liu, and Junhui Liu. 2019. Faspell: A fast, adaptable, simple, powerful chinese spell checker based on dae-decoder paradigm. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 160–169.

Li Huang, Junjie Li, Weiwei Jiang, Zhiyu Zhang, Minchuan Chen, Shaojun Wang, and Jing Xiao. 2021. Phmospell: Phonological and morphological knowledge guided chinese spelling check. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5958–5967.

Tuo Ji, Hang Yan, and Xipeng Qiu. 2021. Spellbert: A lightweight pretrained model for chinese spelling check. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3544–3551.

Ying Jiang, Tong Wang, Tao Lin, Fangjie Wang, Wenting Cheng, Xiaofei Liu, Chenghui Wang, and Weijian Zhang. 2012. A rule based chinese spelling and grammar detection system utility. In *2012 International Conference on System Science and Engineering (ICSSE)*, pages 437–440. IEEE.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547.

Nora Kassner and Hinrich Schütze. 2020. Bertknn: Adding a knn search component to pretrained language models for better qa. *arXiv preprint arXiv:2005.00766*.

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Nearest neighbor machine translation. *arXiv preprint arXiv:2010.00710*.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2019. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*.

Huayang Li, Yixuan Su, Deng Cai, Yan Wang, and Lemao Liu. 2022a. A survey on retrieval-augmented text generation. *arXiv preprint arXiv:2202.01110*.

Jiahao Li, Quan Wang, Zhendong Mao, Junbo Guo, Yanyan Yang, and Yongdong Zhang. 2022b. Improving chinese spelling check by character pronunciation prediction: The effects of adaptivity and granularity. *arXiv preprint arXiv:2210.10996*.

Yinghui Li, Qingyu Zhou, Yangning Li, Zhongli Li, Ruiyang Liu, Rongyi Sun, Zizhen Wang, Chao Li, Yunbo Cao, and Hai-Tao Zheng. 2022c. The past mistake is the future wisdom: Error-driven contrastive probability optimization for chinese spell checking. *arXiv preprint arXiv:2203.00991*.

C-L Liu, M-H Lai, K-W Tien, Y-H Chuang, S-H Wu, and C-Y Lee. 2011. Visually and phonologically similar characters in incorrect chinese words: Analyses, identification, and applications. *ACM Transactions on Asian Language Information Processing (TALIP)*, 10(2):1–39.

Shulin Liu, Shengkang Song, Tianchi Yue, Tao Yang, Huihui Cai, TingHao Yu, and Shengli Sun. 2022. Craspell: A contextual typo robust approach to improve chinese spelling correction. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3008–3018.

Shulin Liu, Tao Yang, Tianchi Yue, Feng Zhang, and Di Wang. 2021. 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 (Volume 1: Long Papers)*, pages 2991–3000.

Lidia Mangu and Eric Brill. 1997. Automatic rule acquisition for spelling correction. In *ICML*, volume 97, pages 187–194. Citeseer.

Minh Nguyen, Gia H Ngo, and Nancy F Chen. 2020. Domain-shift conditioning using adaptable filtering via hierarchical embeddings for robust chinese spell check. *arXiv preprint arXiv:2008.12281*.

Yuen-Hsien Tseng, Lung-Hao Lee, Li-Ping Chang, and Hsin-Hsi Chen. 2015. Introduction to sighan 2015 bake-off for chinese spelling check. In *Proceedings of the Eighth SIGHAN Workshop on Chinese Language Processing*, pages 32–37.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Baoxin Wang, Wanxiang Che, Dayong Wu, Shijin Wang, Guoping Hu, and Ting Liu. 2021. Dynamic connected networks for chinese spelling check. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2437–2446.

Dingmin Wang, Yan Song, Jing Li, Jialong Han, and Haisong Zhang. 2018. A hybrid approach to automatic corpus generation for chinese spelling check. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2517–2527.

Dingmin Wang, Yi Tay, and Li Zhong. 2019. Confusionset-guided pointer networks for chinese spelling check. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5780–5785.

Jason Weston, Emily Dinan, and Alexander H Miller. 2018. Retrieve and refine: Improved sequence generation models for dialogue. *arXiv preprint arXiv:1808.04776*.

Shih-Hung Wu, Chao-Lin Liu, and Lung-Hao Lee. 2013. Chinese spelling check evaluation at sighan bake-off 2013. In *SIGHAN@ IJCNLP*, pages 35–42. Citeseer.

Bright Xu. 2019. Nlp chinese corpus: Large scale chinese corpus for nlp.

Heng-Da Xu, Zhongli Li, Qingyu Zhou, Chao Li, Zizhen Wang, Yunbo Cao, Heyan Huang, and Xian-Ling Mao. 2021. Read, listen, and see: Leveraging multimodal information helps chinese spell checking. *arXiv preprint arXiv:2105.12306*.

Junjie Yu and Zhenghua Li. 2014. Chinese spelling error detection and correction based on language model, pronunciation, and shape. In *Proceedings of The Third CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 220–223.

Liang-Chih Yu, Lung-Hao Lee, Yuen-Hsien Tseng, and Hsin-Hsi Chen. 2014. Overview of sighan 2014 bake-off for chinese spelling check. In *Proceedings of The Third CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 126–132.

Ruiqing Zhang, Chao Pang, Chuanqiang Zhang, Shuohuan Wang, Zhongjun He, Yu Sun, Hua Wu, and Haifeng Wang. 2021. Correcting chinese spelling errors with phonetic pre-training. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2250–2261.

Shaohua Zhang, Haoran Huang, Jicong Liu, and Hang Li. 2020. Spelling error correction with soft-masked bert. *arXiv preprint arXiv:2005.07421*.

A Statistics of the Datasets

The statistics of the dataset we used to make the datastore in our experiments are shown in Table 5.

Dataset	#Sent	#Error	#Error-pair
SIGHAN13	1000	1217	748
SIGHAN14	1062	769	461
SIGHAN15	1100	703	460
SIGHANTrain	6126	8470	3318
Wang271K	271329	381962	22409

Table 5: Statistics of the SIGHAN (transferred to simplified Chinese) and Wang271K. We report the number of sentences in the datasets (#Sent), the number of misspellings the datasets contains (#Errors) and the number of different kinds of errors (#Error-pair).

B Hyperparameters of the Model

Since we do not need to retrain CSC model, we do not need to set hyperparameters such as epoch, batch size, and learning rate, which brings great convenience. The hyperparameters of the model at the inference time are obtained on the validation set and we show them in Table 6.

Parameter	Value
k	12
n-gram	3
λ	0.45
Temperature	50
W_{value}	(1.68, 0.68, 1.68)
δ	1
seed	17
size of datastore	13254850

Table 6: The hyperparameters of the model, obtained by testing on the validation set.

C More Detailed Results

Some previous work used scores calculated by the official evaluation tool, which are provided along with the datasets [2,3,4], and some work reported results in character-level metrics only. The baseline methods include: 1) **SpellBERT** (Ji et al., 2021) is a lightweight pre-trained model for CSC. 2) **GAD** (Guo et al., 2021) proposes a global attention decoder approach for CSC. 3) **CRASpell** (Liu et al., 2022) proposes a noise modeling module to generate noisy context in training process to improve performance of the CSC model. In order to compare with these works and to facilitate the comparison of later works, our more detailed results on SIGHAN2015 are shown in Table 7 and 8.

D Results with Data Augmentation

To demonstrate the advantages of our approach that allows simple data augmentation and to compare more fairly with SCOPE (Li et al., 2022b) that uses additional data, we also use the wiki2019zh (Xu, 2019) mentioned above, which is licensed under the MIT License. Wiki2019zh corpus consists of one million Chinese Wikipedia articles that are all with no spelling errors. Note that we do not use the confusion set to make pseudo data, but add these correct sentences to the datastore directly.

As shown in Table 9, after performing data augmentation, our model substantially outperforms SCOPE on SIGHAN2013 and SIGHAN2014, while being slightly lower on SIGHAN2015.

In the meantime, there is an overall improvement in the scores compared to our results without data augmentation. It also proves that the simple way of adding the correct sentences directly to the datastore can effectively improve the results of our model.

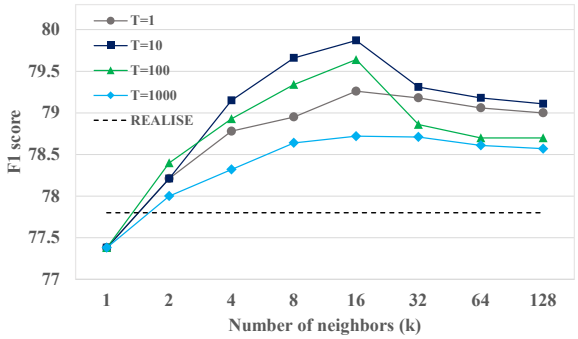

Figure 2: Effect of the number of neighbors retrieved and the softmax temperature on the SIGHAN 2015 test set. The performance of the baseline is marked with a dashed line.

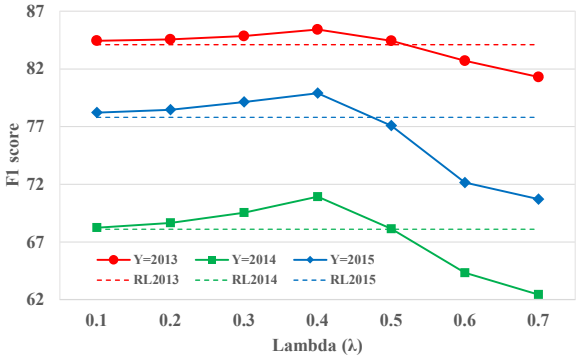

Figure 3: Effect of the interpolation parameter λ on the SIGHAN 2013, 2014 and 2015 test set. The performance of the REALISE in these test sets is marked with a dashed line in the same color.

E Effect of Key Hyperparameters

Effect of the number of neighbors retrieved and the softmax temperature on the SIGHAN 2015 test set is shown in Figure 2. Effect of the interpolation parameter λ on the SIGHAN 2013, 2014 and 2015 test set is shown in Figure 3. For more convenient comparison, we also show the baseline scores in the figures.

F Inference Time

We compare the inference time on SIGHAN2015 test set of CSC model(REALISE) and InfoKNN-CSC conditioned on different batch size. The results are summarized in Table 10.

[2] http://ir.itc.ntnu.edu.tw/lre/sighan7csc.html
[3] http://ir.itc.ntnu.edu.tw/lre/clp14csc.html
[4] http://ir.itc.ntnu.edu.tw/lre/sighan8csc.html

Dataset	Model	Detection-level			Correction-level		
		D-P	D-R	D-F	C-P	C-R	C-F
SIGHAN2015	SpellGCN(Cheng et al., 2020)	77.7	85.6	81.4	96.9	82.9	89.4
	CRASpell(Liu et al., 2022)	83.5	**89.2**	86.3	97.1	**86.6**	91.5
	InfoKNN-CSC (ours)	**88.1**	87.9	**88.0**	**98.6**	85.4	**91.6**

Table 7: The results on SIGHAN2015 of our model and baseline models on character-level metrics, where baseline results are directly from other published paper. Note that CRASpell uses an additional 3 million unlabeled corpus for pre-training, compared to our model.

Dataset	Model	Detection-level			Correction-level		
		D-P	D-R	D-F	C-P	C-R	C-F
SIGHAN2015	SpellGCN(Cheng et al., 2020)	85.9	80.6	83.1	85.4	77.6	81.3
	GAD(Guo et al., 2021)	86.0	80.4	83.1	85.6	77.8	81.5
	SpellBERT(Ji et al., 2021)	87.5	73.6	80.0	87.1	71.5	78.5
	InfoKNN-CSC (ours)	**89.6**	**81.2**	**85.2**	**89.5**	**80.0**	**84.5**

Table 8: The results on SIGHAN2015 of our model and baseline models on sentence-level metrics calculated by SIGHAN official evaluation tools, where baseline results are directly from other published paper.

Dataset	Model	Detection-level			Correction-level		
		D-P	D-R	D-F	C-P	C-R	C-F
SIGHAN2013	SCOPE (Li et al., 2022b)	87.4	**83.4**	85.4	86.3	**82.4**	84.3
	InfoKNN-CSC‡	90.0	82.8	86.3	89.1	82.2	85.6
	InfoKNN-CSC	**90.2**	83.1	**86.5**	**89.2**	82.2	**85.6**
SIGHAN2014	SCOPE (Li et al., 2022b)	70.1	**73.1**	71.6	68.6	**71.5**	70.1
	InfoKNN-CSC‡	72.1	70.6	71.3	71.3	69.8	70.6
	InfoKNN-CSC	**72.3**	71.0	**71.7**	**71.4**	70.0	**70.7**
SIGHAN2015	SCOPE (Li et al., 2022b)	81.1	**84.3**	**82.7**	79.2	**82.3**	80.7
	InfoKNN-CSC‡	81.1	81.3	81.2	79.9	80.1	80.0
	InfoKNN-CSC	**81.4**	81.7	81.5	**80.7**	81.0	**80.9**

Table 9: The results of our model and SCOPE model after performing data augmentation. The result marked with "‡" is the original result obtained without data augmentation. Note that SCOPE is a concurrent work with us.

ms/sent	K	batch=1	batch=16	batch=32	batch=64	batch=128
CSC	-	86.6	60.6	54.7	41.9	39.1
InfoKNN-CSC	4	97.6(×1.13)	64.9(×1.07)	56.6(×1.03)	43.6(×1.04)	40.5(×1.04)
	8	103.7(×1.20)	68.5(×1.13)	57.3×(1.05)	45.6(×1.09)	40.9(×1.05)
	16	105.4(×1.22)	69.7(×1.15)	58.4(×1.07)	47.0(×1.12)	42.8(×1.09)
	32	106.5(×1.23)	70.4(×1.16)	59.7(×1.09)	48.4(×1.16)	45.3(×1.15)

Table 10: Inference time of REALISE and InfoKNN-CSC. All results are tested on 112 cores Intel(R) Xeon(R) Gold 6330 CPU 2.00GHz with a A40-48GB GPU

G Limitations

Since our study focuses on exploring the augmentation of kNN on CSC, we don't attempt the improved version of other kNN method, such as adaptive kNN or fast kNN, which may lead to further improvement. In addition, due to the common problem of retrieval-based methods, many samples need to be stored, which can take up lots of disk space. Specifically, the key stored takes up about 50G of disk space. But after processing by the FAISS (Johnson et al., 2019), the total size of the datastore is less than 1G.