SoftCorrect: Error Correction with Soft Detection for Automatic Speech Recognition

Yichong Leng[1*], Xu Tan[2], Wenjie Liu[3], Kaitao Song[2], Rui Wang[2]
Xiang-Yang Li[1], Tao Qin[2], Ed Lin[3], Tie-Yan Liu[2]
[1]University of Science and Technology of China
[2]Microsoft Research Asia
[3]Microsoft Azure Speech
[1]lyc123go@mail.ustc.edu.cn,xiangyangli@ustc.edu.cn
[2]{xuta,kaitaosong,ruiwa,taoqin,tyliu}@microsoft.com [3]{liwenjie,edlin}@microsoft.com

Abstract

Error correction in automatic speech recognition (ASR) aims to correct those incorrect words in sentences generated by ASR models. Since recent ASR models usually have low word error rate (WER), to avoid affecting originally correct tokens, error correction models should only modify incorrect words, and therefore detecting incorrect words is important for error correction. Previous works on error correction either *implicitly* detect error words through target-source attention or CTC (connectionist temporal classification) loss, or *explicitly* locate specific deletion/substitution/insertion errors. However, implicit error detection does not provide clear signal about which tokens are incorrect and explicit error detection suffers from low detection accuracy. In this paper, we propose SoftCorrect with a soft error detection mechanism to avoid the limitations of both explicit and implicit error detection. Specifically, we first detect whether a token is correct or not through a probability produced by a dedicatedly designed language model, and then design a constrained CTC loss that only duplicates the detected incorrect tokens to let the decoder focus on the correction of error tokens. Compared with implicit error detection with CTC loss, SoftCorrect provides explicit signal about which words are incorrect and thus does not need to duplicate every token but only incorrect tokens; compared with explicit error detection, SoftCorrect does not detect specific deletion/substitution/insertion errors but just leaves it to CTC loss. Experiments on AISHELL-1 and Aidatatang datasets show that SoftCorrect achieves 26.1% and 9.4% CER reduction respectively, outperforming previous works by a large margin, while still enjoying fast speed of parallel generation.

Introduction

Correction (Cucu et al. 2013; D'Haro and Banchs 2016; Anantaram et al. 2018; Du et al. 2022) has been widely used in automatic speech recognition (ASR) to refine the output sentences of ASR systems to reduce word error rate (WER). Considering the error rate of the sentences generated by ASR is usually low (e.g., $<10\%$, which means only a small proportion of tokens are incorrect and need correction), how to accurately detect errors is important for correction (Leng et al. 2021b). Otherwise, correct tokens may be changed by mistake, or error tokens cannot be corrected. Previous works conduct error detection in different ways: 1) Implicit error detection, where the errors are not explicitly detected but embedded in the correction process. For example, Liao et al. (2020); Mani et al. (2020); Wang et al. (2020); Zhu et al. (2021) adopt an encoder-decoder based autoregressive correction model with a target-source (decoder-encoder) attention (Vaswani et al. 2017); Gu and Kong (2021) duplicate the source tokens several times and leverage a CTC (connectionist temporal classification) loss (Graves et al. 2006), where the target-source alignments learnt in decoder-encoder attention or CTC paths play a role of implicit error detection. 2) Explicit error detection, where the specific deletion/substitution/insertion errors are detected out explicitly. For example, Leng et al. (2021b,a); Du et al. (2022); Shen et al. (2022) rely on the predicted duration to determine how many target tokens each source token should be corrected to (e.g., 0 stands for deletion error, 1 stands for no change or substitution error, ≥ 2 stands for insertion error).

Implicit error detection enjoys the advantage of the flexibility of model learning but suffers from the limitation that it does not provide clear signal for model training about which tokens are incorrect. In the contrast, explicit error detection enjoys the advantage of clear signal but suffers from the limitation that it requires precise error patterns and thus new error will be introduced once error detection is not accurate. For example, if a substitution error is predicted as an insertion error by explicit error detection, then the model cannot correct this error but introduce new error by inserting a wrong token. A natural question arises: can we design a better error detection mechanism that inherits the advantages of both implicit and explicit error detection and avoids their limitations?

To answer this question, in this paper, we propose a soft error detection mechanism with an error detector (encoder) and an error corrector (decoder). Specifically, we dedicatedly design a language model as the encoder to determine whether a token is correct or not and design a constrained CTC loss on the decoder to only focus on correcting the de-

[*]This work was conducted at Microsoft. Corresponding author: Xu Tan, xuta@microsoft.com

tected error tokens (focused error detection):

- Instead of predicting correction operations such as deletion, substitution and insertion (Leng et al. 2021b,a; Du et al. 2022), we only detect whether a token is correct or not. To this end, we can either use a binary classification (Fang et al. 2022) or the probability from language model for error detection. We choose the latter one since the probability from a language model contains more knowledge in language understanding and vocabulary space (Hinton et al. 2015; Gou et al. 2021) than a simple binary classification. However, previous methods for language modeling such as left-to-right language modeling (e.g., GPT (Brown et al. 2020)) and bidirectional language modeling (e.g., BERT (Devlin et al. 2019)) are not suitable in our scenario: 1) left-to-right language models like GPT only leverage unidirectional context information (e.g., left), which cannot provide enough context information for accurate probability estimation; 2) bidirectional language models like BERT can leverage bidirectional context information, but it needs N passes (where N corresponds to the number of tokens) (Salazar et al. 2020) to estimate the probability of all the N tokens in a sentence, which cannot satisfy the fast speed requirement for error detection. In this paper, we train the encoder with a novel language model loss, to output probabilities effectively and efficiently to detect error tokens in source sentence.
- Instead of duplicating all the source tokens multiple times in CTC loss, we only duplicate the incorrect tokens detected (as indicated by the probabilities from the encoder trained with our novel language model loss) and use a constrained CTC loss to let the decoder focus on the correction of these duplicated error tokens, resulting in a focused error correction. Compared with the standard CTC loss that duplicates all the tokens, our constrained CTC loss provides clear signals about which part of tokens should be corrected.

Furthermore, previous works (Weng et al. 2020; Liu et al. 2018) have shown that the multiple candidates generated by ASR beam search can be leveraged to verify the correctness of tokens (Leng et al. 2021a) in each candidate. To further improve correction accuracy, we take multiple candidates from ASR beam search as encoder input. Accordingly, the error detection in the encoder contains two steps, i.e., first selecting a better candidate from multiple candidates (equivalent to detect which candidates are likely to be incorrect) for further correction, and then detecting which tokens are likely to be incorrect in the selected candidate. The contributions of this paper are summarized as follows:

- We propose SoftCorrect with a soft error detection mechanism for ASR error correction to inherit the advantages of both explicit and implicit error detection and avoid their limitations.
- We design a novel language model loss for encoder to enable error detection and a constrained CTC loss for the decoder to focus on the tokens that are detected as errors.
- Experimental results on AISHELL-1 and Aidatatang datasets demonstrate that SoftCorrect achieves 26.1% and 9.4% CER reduction respectively, while still enjoying fast error correction with parallel generation.

Background

Error Correction for ASR

Error Correction Models Error correction is widely used in ASR systems (Shivakumar et al. 2018; Hu et al. 2020) to reduce word error rate. Error correction models usually take the sentences outputted by ASR systems as input and generate corrected sentences, and have evolved from early statistic machine translation models (Cucu et al. 2013; D'Haro and Banchs 2016), to later neural-network based autoregressive models (Tanaka et al. 2018; Liao et al. 2020; Wang et al. 2020), and to recent non-autoregressive models (Leng et al. 2021b,a; Du et al. 2022). Non-autoregressive error correction models generate sentences in parallel with the help of a duration predictor (Gu et al. 2018) to predict the number of tokens that each input token can be corrected to, which achieve much faster inference speed than autoregressive counterparts and approximate correction accuracy, making it suitable for online deployment.

Multiple Candidates Recent works (Zhu et al. 2021; Liu et al. 2018; Imamura and Sumita 2017; Weng et al. 2020) show that the multiple candidates generated by ASR beam search can have *voting effect* (Leng et al. 2021a), which can be beneficial for both autoregressive and non-autoregressive correction models. They first align the multiple candidates to the same length using multi-candidate alignment algorithm based on the token-level similarity and phoneme-level similarity, and take the aligned candidates as encoder input. SoftCorrect also leverages multiple candidates since the difference of beam search results can show the uncertainty of ASR model and give clues about potential error tokens.

Error Detection by Target-Source Alignments

Error detection can be achieved via the alignments between the target (correct) sentence and the source (incorrect) sentence, which can be either explicit or implicit.

Explicit Alignment By explicitly aligning the source and target sequences together with edit distance (Leng et al. 2021b), we can obtain the number of target tokens (duration) aligned with each source token and train a duration predictor. Thus, we can detect insertion, deletion and substitution error with corresponding duration (e.g., 0 stands for deletion error, 1 stands for no change or substitution error, ≥ 2 stands for insertion error). However, the duration predictor is hard to optimize precisely and thus new error will be introduced once duration prediction is not accurate.

Implicit Alignment Errors can be "detected" via implicit alignment between target and source sequences. For example, Transformer (Vaswani et al. 2017) based autoregressive models embed the target-source alignment in decoder-encoder attention (Wang et al. 2020; Zhu et al. 2021), and CTC-based models (Libovickỳ and Helcl 2018; Saharia et al. 2020; Majumdar et al. 2021) leverage a CTC loss (Graves et al. 2006) to align target with duplicated

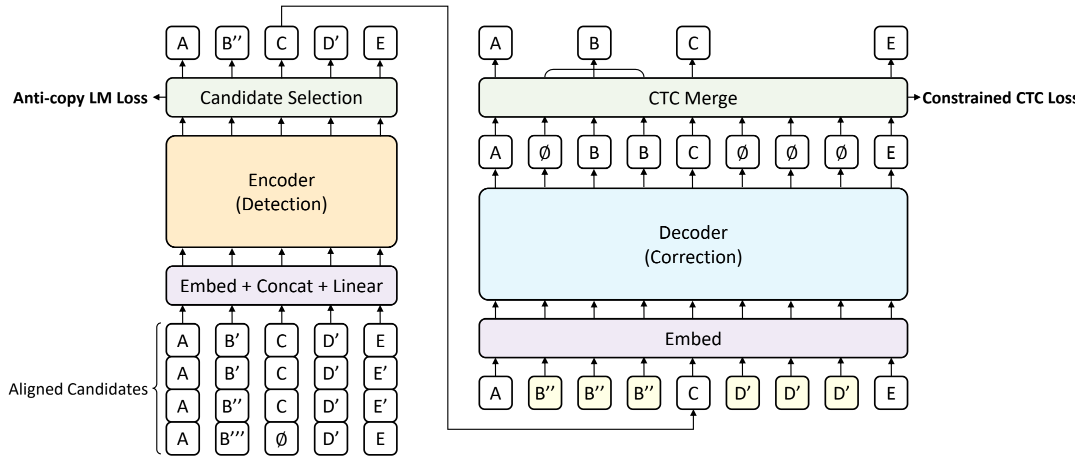

Figure 1: Overview of SoftCorrect. We use $A\ B\ C\ E$ to represent the ground-truth tokens, while $B'\ B''\ B'''\ D'\ E'$ to represent incorrect tokens. We use ϕ to represent blank token for alignment purpose only, which is leveraged in both multi-candidate alignment and CTC alignment. In this case, the ground-truth sentence is $ABCE$, while the 4 candidates are $AB'CD'E$, $AB'CD'E'$, $AB''CD'E'$, and $AB'''D'E$, respectively. The selected candidate is $AB''CD'E$, where B'' and D' are detected as incorrect tokens and duplicated when fed into decoder.

source implicitly. A desirable property of CTC loss is that it enables parallel error correction, without the need of duration prediction and with more flexibility during correction. Thus, in this paper, we adopt the CTC based solution but enhance it with a soft error detection mechanism.

SoftCorrect

System Overview

As shown in Figure 1, SoftCorrect consists of an error detector (the encoder) and a focused error corrector (the decoder). We introduce the whole system step by step:

- Motivated by the voting effect in multiple candidates for error detection, we leverage multiple candidates from ASR beam search. We first align these candidates to the same length following Leng et al. (2021a). The aligned candidates are shown in the bottom left of Figure 1. The aligned candidates are converted into token embeddings, concatenated along the position and fed into a linear layer.
- The detector is a standard Transformer Encoder (Vaswani et al. 2017) which takes the output of the previous step as input, and generates a probability for each token in each candidate. Specifically, the output hidden of the encoder is multiplied with a token-embedding matrix to generate a probability distribution over the whole vocabulary. For example, the output probability distribution in the last position in Figure 1 can provide the probability for token E and E' simultaneously. Since ASR usually has low WER, to prevent encoder from learning trivial copy, we propose an anti-copy language model loss to train the encoder to output this probability distribution.
- Based on the probability, we can choose the token with the highest probability in each position from multiple candidates and obtain a better candidate, which usually contains less errors and thus makes the error detection easier. This step is illustrated as the "Candidate Selection" module in Figure 1 and the selected candidate is $AB''CD'E$. Noted that we conduct *position-wise* selection and the tokens in selected candidate (e.g., B'' and E) can come from different candidates.
- After candidate selection, we combine the probability of each token in selected candidate with its corresponding probability from the ASR model. The error detection score for each token is the weighted linear combination of encoder probability and ASR output probability reflecting the similarity between token pronunciation and audio. A token is detected as incorrect when the combined probability is lower than a threshold (Huang and Peng 2019).
- The corrector (decoder) takes the generated candidate as input and outputs refined tokens. It is trained with a constrained CTC loss, which learns to only modify the detected "incorrect" tokens while directly copying remaining tokens to output. Therefore, we only duplicate the incorrect tokens detected in previous step. As shown in the bottom right of Figure 1, the detected and duplicated error tokens are B'' and D'.

In the next subsections, we introduce the details of the anti-copy language model loss in encoder to generate token probability for error detection, and the constrained CTC loss in decoder for focused error correction.

Anti-Copy Language Modeling for Detection

We use the encoder to output a probability for each token in the multiple candidates, where this probability can be used in candidate selection and error detection. Since we need to select better candidate as well as detect errors, the probability from binary classification based error detection method (Omelianchuk et al. 2020) is unsuitable for the lack of knowledge in language understanding and vocabulary space (Hinton et al. 2015; Gou et al. 2021). Indeed,

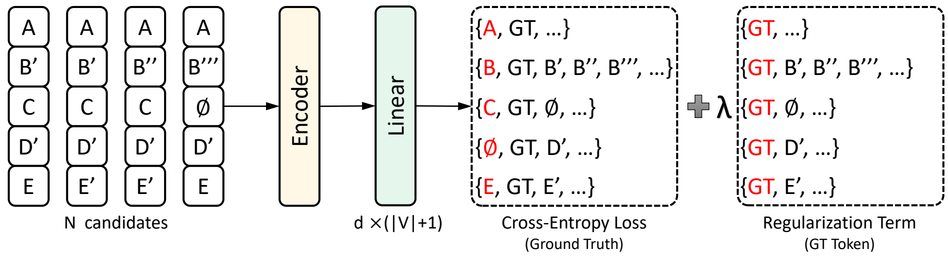

Figure 2: Illustration of the proposed anti-copy language model loss as formulated in Equation 1. The tokens in red color represent the target tokens in each term. The remaining tokens in the vocabulary are represented as "...".

the probability we need is like a kind of language modeling, which determines whether a token is natural/reasonable given the context information. Common language modeling methods are unsuitable for probability estimation since GPT-based methods lack bidirectional context information and BERT-based methods are too slow with N-pass (Salazar et al. 2020) inference. To alleviate these issues, we propose a novel method to not only leverage bidirectional information but also provide fast probability estimation.

A straightforward way is to train the Transformer encoder to predict ground-truth (correct) tokens in each position given multiple aligned candidates as input. In this way, the encoder can learn to output probability to determine whether a token is natural/reasonable given the context information. However, since the ASR systems usually have relatively low WER (e.g., $<$10%), a *large proportion* of tokens in aligned input are consistent (i.e., with the same token) and correct. In this way, directly predicting the ground-truth token in each position would result in trivial copy. For example, the first position in Figure 1 might be a trivial copy since the aligned input tokens are consistent (i.e., all A) and the corresponding target token is also A. This trivial copy issue will cause the model outputting an extreme high probability for the input token on consistent position and thus hurt the ability of encoder on detecting errors on consistent position.

To alleviate this problem, we propose an anti-copy language model loss to prevent learning copy only, which modifies the standard cross-entropy loss by changing its prediction vocabulary and adding a regularization term. Specifically, we add a special GT token (it does not stand for any specific ground-truth token in each position, but just a special symbol) in the vocabulary as shown in Figure 2, and the objective function to train the encoder is as follows.

$$\begin{aligned}\mathcal{L}_{\text{lm}} &= \sum_{t=1}^{N} \frac{\exp(H_t W_{y_t})}{\sum_{i\in\{V+GT\}} \exp(H_t W_i)} \\ &+ \lambda \sum_{t=1}^{N} \frac{\exp(H_t W_{GT})}{\sum_{i\in\{V\setminus y_t+GT\}} \exp(H_t W_i)},\end{aligned} \tag{1}$$

where V represents the original token vocabulary (including a special token ϕ to represent deletion), $V+GT$ represents the original token vocabulary plus GT token, and $V \setminus y_t$ represents the original token vocabulary minus y_t, where y_t is the ground-truth token at position t. $W \in \mathbb{R}^{d\times(|V|+1)}$, $H \in \mathbb{R}^{N\times d}$, where N is the length of aligned candidates and d is the hidden size of the encoder output. $W_{y_t} \in \mathbb{R}^d$ and $W_{GT} \in \mathbb{R}^d$ represent the vector in the softmax matrix that corresponds to token y_t and GT respectively, and H_t represents the hidden vector generated by the encoder at position t. λ is used to balance the regularization term.

The first term in Equation 1 is a cross-entropy loss to predict the ground-truth token over the full vocabulary (including the GT token). The second is a regularization term, which predicts GT token over the full vocabulary without ground-truth token (including the GT token but removing the ground-truth token). The first term is used to encourage the encoder to output ground-truth token for reasonable probability estimation, and the second term is used to alleviate copying the ground-truth token.

There are two intuitions behind the anti-copy loss: 1) The regularization (second) term of anti-copy loss aims to avoid copy-mapping by removing ground-truth token from vocabulary and train the model to predict GT over all vocabulary except the ground-truth token; 2) Since all error tokens are optimized to have lower probability than GT while the correct token is optimized to have higher probability than GT, an advantages of anti-copy loss is that the GT token can serve as a decision boundary between correct token and error token and help better detect the error token based on the probability of language model with anti-copy loss.

Constrained CTC Loss for Correction

As aforementioned, we obtain the constructed candidate from the encoder with detected errors and feed it as the input of the decoder to generate the final corrected result. Since we know which tokens in the selected candidate are correct or incorrect, it is unnecessary to modify all the tokens in the decoder, which can cause larger latency and produce new errors (e.g., a correct token is edited as an incorrect token). Therefore, we propose a constrained CTC loss as shown in Figure 3. We only repeat the incorrect tokens and keep the remaining tokens unchanged for decoder input. Incorrect tokens are repeated three times, which is a common practice for CTC loss (Libovický and Helcl 2018; Saharia et al. 2020). The likelihood of the target sequence y is the probability summation of all possible CTC alignment paths after constraints:

$$\mathcal{L}_{\text{ctc}} = \sum_{z\in\phi'(y)} P(z|x), \tag{2}$$

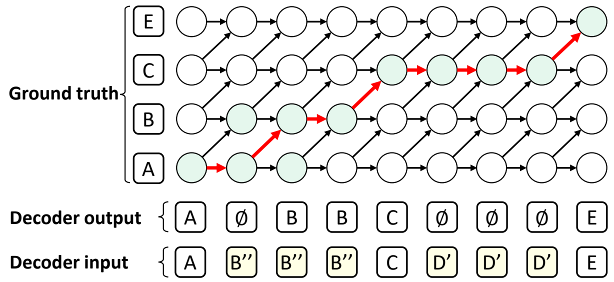

Figure 3: Constrained CTC loss, which only allows dynamic alignment between the output and target for the detected (repeated) tokens (e.g., B'' and D'), but uses fixed alignment for the undetected (not repeated) tokens (e.g., A, C, and E). The nodes in green color represent all possible CTC alignments while the nodes connected with red arrows represent the alignment for the decoder output (just one possible case) shown in this figure.

where x is the decoder input (i.e., the selected candidate with expansion on the "incorrect tokens", just as "decoder input" shown in Figure 3), $\phi'(y)$ represents all the possible alignment paths to generate y in our constrained CTC (different from the standard alignments $\phi(y)$), and z represents one possible CTC path, as shown in the red arrow in Figure 3.

The main difference between our loss and standard CTC loss is the *all possible alignment paths* $\phi'(y)$. As shown in Figure 3, the possible CTC alignment paths $\phi'(y)$ lay only on the shaded (green) nodes, in contrast to the standard CTC where the alignment paths $\phi(y)$ lay on all nodes. As a result, the "correct token" is used as an anchor and cannot be dynamically aligned (the output of anchor token must be the anchor token itself). During the inference, we perform softmax over the possible error tokens, select the best token from each of those positions and then remove duplicates and blank. With the help of the explicit error detection from encoder, we can skip the correction process of the decoder to reduce the system latency if all input tokens are detected as correct token.

Explicit error detector can possibly produce new errors (e.g., some correct tokens are identified as errors) and propagate them to the error corrector. Hence, we need the error corrector to be more robust to the outputs of error detector. Specifically, when training corrector, we randomly select 5% correct tokens and regard them as pseudo error tokens to simulate the mistakes from detector, so that model will not modify these correct tokens during the optimization and can be more robust to the outputs of detector.

Experimental Setup

In this section, we introduce the datasets and the ASR model used for correction, and some previous error correction systems for comparison.

Datasets and ASR Model

We conduct experiments on two Mandarin ASR datasets, AISHELL-1 (Bu et al. 2017) and Aidatatang_200zh, referred to as Aidatatang for short. AISHELL-1 contains 150/10/5-hour speech data for train/ development/test, while Aidatatang contains 140/20/40-hour speech data for train/development/test, respectively.

The ASR model used in our experiments is a state-of-the-art model with Conformer architecture (Gulati et al. 2020), enhanced with SpecAugment (Park et al. 2019) and speed perturbation for data augmentation, and a language model for joint decoding. The hyper-parameters of this ASR model follow the ESPnet codebase (Watanabe et al. 2018)[1][2].

The training, development, and test data for correction models are obtained by using the ASR model to transcribe the corresponding datasets in AISHELL-1 and Aidatatang. Following the common practice in ASR correction (Leng et al. 2021b,a; Du et al. 2022; Zhu et al. 2021), we use 400M unpaired text data to construct a pseudo pretraining dataset for both SoftCorrect and baseline systems.

Baseline Systems

We compare SoftCorrect with the several error correction baselines, including systems using implicit and explicit error detection.

For baselines with implicit error detection, we use: 1) *AR Correct*. A standard autoregressive (AR) encoder-decoder model based on Transformer (Vaswani et al. 2017). 2) *AR N-Best*. Following Zhu et al. (2021), we train an AR Transformer model by taking the aligned multiple candidates from ASR beam search as input and generating correction result.

For baselines with explicit error detection, we use: 1) *FastCorrect*. FastCorrect (Leng et al. 2021b) is a non-autoregressive model for ASR correction, which utilizes token duration to adjust input sentence length to enable parallel decoding. 2) *FastCorrect 2*. Leng et al. (2021a) introduce multiple candidates into non-autoregressive FastCorrect model and achieve state-of-the-art correction accuracy.

Considering SoftCorrect leverages multiple candidates, we also take rescoring method for comparison. The rescoring model is a 12-layer Transformer decoder model and the details of rescoring follow Huang and Peng (2019). Besides, we also combine non-autoregressive correction with rescoring together to construct another two baselines. One is *FC + Rescore* where the outputs of ASR are first corrected by FastCorrect (FC for short) and then rescored, the other is *Rescore + FC* where the ASR outputs are first rescored and then corrected.

Results

In this section, we first compare the character error rate reduction (CERR) and latency of SoftCorrect with baselines, and then conduct ablation studies to verify the effectiveness of several designs in SoftCorrect, including anti-copy language model loss and constrained CTC loss. Besides, we conduct some analyses to show the advantages of SoftCorrect on the ability of error detection and error correction over baseline systems.

[1]github.com/espnet/espnet/tree/master/egs/aishell
[2]github.com/espnet/espnet/tree/master/egs/aidatatang_200zh

Model	AISHELL-1				Aidatatang				Latency (ms/sent)	
	Test		Dev		Test		Dev			
	CER	CERR	CER	CERR	CER	CERR	CER	CERR	GPU	CPU
No Correction	4.83	-	4.46	-	4.47	-	3.82	-	-	-
Implicit error detection baselines										
AR Correct	4.07	15.73	3.79	15.02	4.39	1.79	3.74	2.09	119.0 (1.0×)	485.5 (1.0×)
AR N-Best	3.94	18.43	3.68	17.49	4.70	-5.15	4.06	-6.28	121.6 (1.0×)	495.8 (1.0×)
Explicit error detection baselines										
FastCorrect	4.16	13.87	3.89	12.78	4.47	0.00	3.82	0.00	16.2 (7.2×)	92.1 (5.3×)
FastCorrect 2	4.11	14.91	3.78	15.25	4.59	-2.68	4.02	-5.24	23.1 (5.2×)	114.6 (4.2×)
Other baselines										
Rescore	4.02	16.77	3.74	16.14	4.29	4.03	3.64	4.71	48.8 (2.4×)	256.0 (1.9×)
Rescore + FC	3.69	23.60	3.48	21.97	4.33	3.13	3.68	3.66	65.0 (1.8×)	348.1 (1.4×)
FC + Rescore	3.58	25.88	3.40	23.77	4.29	4.03	3.63	4.97	113.6 (1.0×)	624.4 (0.8×)
SoftCorrect	**3.57**	**26.09**	**3.40**	**23.77**	**4.05**	**9.40**	**3.44**	**9.95**	17.0 (7.0×)	97.4 (5.0×)

Table 1: The correction accuracy and inference latency of different systems. We report the character error rate (CER) and character error rate reduction (CERR) on test and development sets of the two datasets, and report the inference latency measured on NVIDIA V100 GPU or "Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz" CPU on the test set of AISHELL-1. "FC" stands for FastCorrect in other baselines.

Accuracy and Latency

We report the correction accuracy and inference latency of different systems in Table 1. We have several observations:

1) Compared with non-autoregressive baselines FastCorrect and FastCorrect 2 with explicit error detection, SoftCorrect achieves larger CERR while still enjoying low latency, which demonstrates the effectiveness of SoftCorrect with soft error detection.

2) Compared with autoregressive baselines AR and AR N-Best with implicit error detection, SoftCorrect is the first non-autoregressive system achieving larger CERR than them.

3) Compared with combined systems (Rescore+FC and FC+Rescore), SoftCorrect achieves slightly better CERR but much faster inference speed (7x speedup over FC + Rescore in terms of latency).

4) On Aidatatang dataset, the errors are hard to detect and some previous correction baselines fail to reduce CER, while SoftCorrect still achieves more than 9% CERR, demonstrating the advantage of our soft detection mechanism by first detecting error tokens and then focusing on correcting errors tokens. Since ASR results have few error tokens, accurate error detection is necessary to avoid mistaking originally correct tokens or missing originally incorrect tokens. As a result, our design of soft error detection achieves good results for ASR correction.

Ablation Studies

We conduct ablation studies to verify the effectiveness of our soft detection with encoder for error detection and decoder for focused correction, as shown in Table 2. We introduce these studies as follows:

- We first remove our anti-copy language model loss and simply use standard cross-entropy loss to train the encoder (Setting ID 2), resulting in an inferior accuracy due to the model may learn to copy the ground-truth token, hurting the error detection ability.
- Since there are some popular methods to model token-level probability such as BERT (Devlin et al. 2019) and GPT (Brown et al. 2020), we apply BERT-style and GPT-style training loss on the encoder to detect errors in the aligned multiple candidates, as shown in ID 3 and 4. Moreover, we also train the encoder to perform detection on each token of each candidate with a binary classification loss (ID 5). The poor results of GPT-style training shows that bi-directional information is necessary for error-detection. The BERT-style training or binary classification achieves lower accuracy than SoftCorrect, showing the effectiveness of our anti-copy language model loss.
- When removing the constraint on the CTC loss (ID 6), the correction accuracy is lower, which demonstrates the advantage of only focusing on correcting the detected error tokens. The results also show that the error detector is reliable because the attempt on modifying tokens that are detected to be right (Setting ID 6) only leads to worse accuracy.

Method Analyses

We compare the error detection and correction ability of SoftCorrect with previous autoregressive and non-autoregressive baselines. We measure the error detection ability using the precision (P_{det}) and recall (R_{det}) of that an error token is detected as error, and measure the error correction ability using the precision (P_{cor}) of that an error

ID	Setting		AISHELL-1				Aidatatang			
			Test		Dev		Test		Dev	
			CER ↓	CERR ↑	CER ↓	CERR ↑	CER↓	CERR ↑	CER↓	CERR ↑
1	SoftCorrect		3.57	26.09	3.40	23.77	4.05	9.40	3.44	9.95
2	Encoder	CE loss	3.77	21.94	3.60	19.28	4.21	5.82	3.56	6.81
3		BERT-style	3.93	18.63	3.73	16.37	4.14	7.38	3.52	7.85
4		GPT-style	4.76	1.45	4.36	2.24	4.41	1.34	3.78	1.05
5		Binary	3.98	17.60	3.75	15.92	4.26	4.70	3.62	5.24
6	Decoder	- Constraint	3.95	18.22	3.69	17.26	4.14	7.38	3.52	7.85
7	No Correction		4.83	-	4.46	-	4.47	-	3.82	-

Table 2: Ablation studies on the designs in SoftCorrect, including some variants of the anti-copy language model loss for encoder and the constrained CTC loss for decoder. “CE loss” means using standard cross-entropy loss to predict ground truth, “BERT-style” refers to using BERT model to estimate the probability of each token via N-pass (Salazar et al. 2020). “GPT-style” refers to using left-to-right language model to estimate the probability of each token and “Binary” refers to using binary classification to detect errors.

Model	AISHELL-1					Aidatatang				
	P_{det}	R_{det}	$F1_{det}$	P_{cor}	CERR	P_{det}	R_{det}	$F1_{det}$	P_{cor}	CERR
Implicit error detection baselines										
AR Correct	**84.56**	33.00	54.73	64.73	15.73	73.55	14.32	35.34	48.05	1.79
AR N-Best	76.03	45.13	54.96	**72.29**	18.43	57.98	**32.18**	32.78	56.55	-5.15
Explicit error detection baselines										
FastCorrect	83.72	34.54	50.10	59.84	13.87	69.78	9.78	29.78	42.68	0.0
FastCorrect 2	80.58	32.54	56.51	70.13	14.91	60.50	23.40	32.35	53.48	-2.68
SoftCorrect	84.06	**49.71**	**59.94**	71.30	**26.09**	**80.52**	25.29	**49.32**	**61.25**	**9.40**

Table 3: Comparison of different systems in terms of error detection and correction ability. P_{det}, R_{det}, and $F1_{det}$ represent the precision, recall, and F1 score of error detection. P_{cor} represents the precision of correction on error tokens. The character error rate reduction (CERR) is also shown.

token is corrected to its corresponding ground-truth token. For autoregressive models that use implicit error detection, we assume a model detect a source token as error token if the model edits that token to another token.

As shown in Table 3, we can observe that: 1) on both datasets, SoftCorrect achieves better P_{det}, R_{det}, and P_{cor} than non-autoregressive baselines with explicit error detection, which shows SoftCorrect has a stronger ability on error detection and correction; 2) Compared with autoregressive baselines using implicit error detection, SoftCorrect performs better on balancing the precision and recall of error detection (higher $F1_{det}$), which verifies the necessity of soft error detection. 3) The errors in Aidatatang dataset is hard to detect, which cannot be handled with implicit error detection or duration-based explicit error detection. On this dataset, previous method may mistake a correct token which introduces new error, or miss an incorrect token. In contrast, the precision or recall of the detection in SoftCorrect is higher, demonstrating the advantage of soft error detection. Moreover, with high-accurate error detection, constrained CTC loss makes the error correction more focused and thus easier, resulting in the higher P_{cor} of SoftCorrect.

Conclusion

In this paper, we design a soft error detection mechanism for ASR error correction, which consists of an encoder for error detection and a decoder for focused error correction. Considering error detection is important for ASR error correction and previous works using either explicit or implicit error detection suffer from some limitations, we propose SoftCorrect with a soft error detection mechanism. Specifically, we design an anti-copy language model loss to enable the encoder to select a better candidate from multiple input candidates and detect errors in the selected candidate, and design a constrained CTC loss to help decoder focus on correcting detected error tokens while keeping undetected tokens unchanged. Experimental results show that SoftCorrect achieves much larger CER reduction compared with previous explicit and implicit error detection methods in ASR error correction, while still enjoying fast inference speed.

References

Anantaram, C.; Sangroya, A.; Rawat, M.; and Chhabra, A. 2018. Repairing ASR output by Artificial Development and Ontology based Learning. In *IJCAI*, 5799–5801.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D. M.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*.

Bu, H.; Du, J.; Na, X.; Wu, B.; and Zheng, H. 2017. Aishell-1: An open-source mandarin speech corpus and a speech recognition baseline. In *20th Conference of the Oriental Chapter of the International Coordinating Committee on Speech Databases and Speech I/O Systems and Assessment (O-COCOSDA)*.

Cucu, H.; Buzo, A.; Besacier, L.; and Burileanu, C. 2013. Statistical Error Correction Methods for Domain-Specific ASR Systems. In Dediu, A.-H.; Martín-Vide, C.; Mitkov, R.; and Truthe, B., eds., *International Conference on Statistical Language and Speech Processing*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*, 4171–4186.

Du, J.; Pu, S.; Dong, Q.; Jin, C.; Qi, X.; Gu, D.; Wu, R.; and Zhou, H. 2022. Cross-Modal ASR Post-Processing System for Error Correction and Utterance Rejection. In *arXiv preprint arXiv:2201.03313*.

D'Haro, L. F.; and Banchs, R. E. 2016. Automatic correction of ASR outputs by using machine translation. In *INTERSPEECH*.

Fang, Z.; Zhang, R.; He, Z.; Wu, H.; and Cao, Y. 2022. Non-Autoregressive Chinese ASR Error Correction with Phonological Training. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 5907–5917.

Gou, J.; Yu, B.; Maybank, S. J.; and Tao, D. 2021. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6): 1789–1819.

Graves, A.; Fernández, S.; Gomez, F.; and Schmidhuber, J. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, 369–376.

Gu, J.; Bradbury, J.; Xiong, C.; Li, V. O. K.; and Socher, R. 2018. Non-Autoregressive Neural Machine Translation. In *6th International Conference on Learning Representations, ICLR*, 1–12.

Gu, J.; and Kong, X. 2021. Fully Non-autoregressive Neural Machine Translation: Tricks of the Trade. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 120–133. Online: Association for Computational Linguistics.

Gulati, A.; Qin, J.; Chiu, C.-C.; Parmar, N.; Zhang, Y.; Yu, J.; Han, W.; Wang, S.; Zhang, Z.; Wu, Y.; et al. 2020. Conformer: Convolution-augmented transformer for speech recognition. In *INTERSPEECH*.

Hinton, G.; Vinyals, O.; Dean, J.; et al. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7).

Hu, K.; Sainath, T. N.; Pang, R.; and Prabhavalkar, R. 2020. Deliberation model based two-pass end-to-end speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Huang, H.; and Peng, F. 2019. An Empirical Study of Efficient ASR Rescoring with Transformers. *arXiv preprint arXiv:1910.11450*.

Imamura, K.; and Sumita, E. 2017. Ensemble and Reranking: Using Multiple Models in the NICT-2 Neural Machine Translation System at WAT2017. In *Proceedings of the 4th Workshop on Asian Translation, WAT@IJCNLP*, 127–134.

Leng, Y.; Tan, X.; Wang, R.; Zhu, L.; Xu, J.; Liu, W.; Liu, L.; Li, X.-Y.; Qin, T.; Lin, E.; et al. 2021a. FastCorrect 2: Fast Error Correction on Multiple Candidates for Automatic Speech Recognition. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, 4328–4337.

Leng, Y.; Tan, X.; Zhu, L.; Xu, J.; Luo, R.; Liu, L.; Qin, T.; Li, X.-Y.; Lin, E.; and Liu, T.-Y. 2021b. FastCorrect: Fast Error Correction with Edit Alignment for Automatic Speech Recognition. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*.

Liao, J.; Eskimez, S. E.; Lu, L.; Shi, Y.; Gong, M.; Shou, L.; Qu, H.; and Zeng, M. 2020. Improving readability for automatic speech recognition transcription. *CoRR*, abs/2004.04438.

Libovickỳ, J.; and Helcl, J. 2018. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. *arXiv preprint arXiv:1811.04719*.

Liu, Y.; Zhou, L.; Wang, Y.; Zhao, Y.; Zhang, J.; and Zong, C. 2018. A Comparable Study on Model Averaging, Ensembling and Reranking in NMT. In *Natural Language Processing and Chinese Computing - 7th CCF International Conference, NLPCC, Part II*, volume 11109, 299–308.

Majumdar, S.; Balam, J.; Hrinchuk, O.; Lavrukhin, V.; Noroozi, V.; and Ginsburg, B. 2021. Citrinet: Closing the Gap between Non-Autoregressive and Autoregressive End-to-End Models for Automatic Speech Recognition. In *arXiv preprint arXiv:2104.01721*.

Mani, A.; Palaskar, S.; Meripo, N. V.; Konam, S.; and Metze, F. 2020. ASR error correction and domain adaptation using machine translation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Omelianchuk, K.; Atrasevych, V.; Chernodub, A.; and Skurzhanskyi, O. 2020. GECToR – Grammatical Error Correction: Tag, Not Rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, 163–170.

Park, D. S.; Chan, W.; Zhang, Y.; Chiu, C.-C.; Zoph, B.; Cubuk, E. D.; and Le, Q. V. 2019. Specaugment: A simple data augmentation method for automatic speech recognition. In *INTERSPEECH*.

Saharia, C.; Chan, W.; Saxena, S.; and Norouzi, M. 2020. Non-autoregressive machine translation with latent alignments. *arXiv preprint arXiv:2004.07437*.

Salazar, J.; Liang, D.; Nguyen, T. Q.; and Kirchhoff, K. 2020. Masked Language Model Scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL*, 2699–2712.

Shen, K.; Leng, Y.; Tan, X.; Tang, S.; Zhang, Y.; Liu, W.; and Lin, E. 2022. Mask the Correct Tokens: An Embarrassingly Simple Approach for Error Correction. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2022*.

Shivakumar, P. G.; Li, H.; Knight, K.; and Georgiou, P. G. 2018. Learning from Past Mistakes: Improving Automatic Speech Recognition Output via Noisy-Clean Phrase Context Modeling. *CoRR*, abs/1802.02607.

Tanaka, T.; Masumura, R.; Masataki, H.; and Aono, Y. 2018. Neural Error Corrective Language Models for Automatic Speech Recognition. In *INTERSPEECH*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30 (NIPS)*.

Wang, H.; Dong, S.; Liu, Y.; Logan, J.; Agrawal, A. K.; and Liu, Y. 2020. ASR Error Correction with Augmented Transformer for Entity Retrieval. In *Interspeech*, 1550–1554.

Watanabe, S.; Hori, T.; Karita, S.; Hayashi, T.; Nishitoba, J.; Unno, Y.; Enrique Yalta Soplin, N.; Heymann, J.; Wiesner, M.; Chen, N.; Renduchintala, A.; and Ochiai, T. 2018. ESPnet: End-to-End Speech Processing Toolkit. In *INTERSPEECH*.

Weng, Y.; Miryala, S. S.; Khatri, C.; Wang, R.; Zheng, H.; Molino, P.; Namazifar, M.; Papangelis, A.; Williams, H.; Bell, F.; et al. 2020. Joint contextual modeling for asr correction and language understanding. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6349–6353. IEEE.

Zhu, L.; Liu, W.; Liu, L.; and Lin, E. 2021. IMPROVING ASR ERROR CORRECTION USING N-BEST HYPOTHESES. In *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*.