Towards Contextual Spelling Correction for Customization of End-to-end Speech Recognition Systems

Xiaoqiang Wang, Yanqing Liu, Jinyu Li, Veljko Miljanic, Sheng Zhao, Hosam Khalil

***Abstract*—Contextual biasing is an important and challenging task for end-to-end automatic speech recognition (ASR) systems, which aims to achieve better recognition performance by biasing the ASR system to particular context phrases such as person names, music list, proper nouns, etc. Existing methods mainly include contextual LM biasing and adding bias encoder into end-to-end ASR models. In this work, we introduce a novel approach to do contextual biasing by adding a contextual spelling correction model on top of the end-to-end ASR system. We incorporate contextual information into a sequence-to-sequence spelling correction model with a shared context encoder. The proposed model includes two different mechanisms: autoregressive (AR) and non-autoregressive (NAR). We also propose filtering algorithms to handle large-size context lists, and performance balancing mechanisms to control the biasing degree of the model. The proposed model is a general biasing solution which is domain-insensitive and can be adopted in different scenarios. Experiments show that the proposed method achieves as much as 51% relative word error rate (WER) reduction over ASR system and outperforms traditional biasing methods. Compared to the AR solution, the NAR model reduces model size by 43.2% and speeds up inference by 2.1 times.**

***Index Terms*—speech recognition, contextual spelling correction, contextual biasing, non-autoregressive.**

I. Introduction

In recent years, end-to-end (E2E) ASR systems [1] have obtained significant improvements and achieved performance comparable to traditional hybrid systems [2, 3]. Some representative works include Attention-based Encoder-Decoder (AED) [4–6], recurrent neural network Transducer (RNN-T) [7–10] and transformer transducer (T-T) [11–13]. However, it's still challenging for E2E models to incorporate contextual information which is dynamic and domain related. Such information may be person names in a personal assistant, commonly used terms in a specific domain, proper nouns and so on. E2E ASR systems may perform poorly when this context information is not covered by the training set or it is pronounced similarly to other terms.

Prior works to customize E2E ASR systems with contextual knowledge can be broadly classified into two categories. The first one is incorporating an external contextual language model (LM) into the E2E decoding framework to bias the recognition results towards context phrase list, which is generally implemented by adopting shallow fusion with a contextual finite state transducer (FST) [14–18]. The second category is adding a context encoder which incorporates contextual information into E2E ASR systems [16, 19, 20]. This method conducts contextual biasing in an E2E manner. However, it changes the source ASR model and [16] also reported scalability issues with large biasing phrase list.

Different from traditional methods, we propose to do contextual biasing on the ASR output with a contextual spelling correction model. We aim to make it an efficient, robust and general solution as a post processing "plug-in" module for E2E ASR customization. To achieve this, an autoregressive (AR) and a non-autoregressive (NAR) contextual spelling correction model [21] are proposed. The AR design, denoted as Contextual Spelling Correction (CSC), adds an additional context encoder into an AR E2E spelling correction model, which contains a text encoder, a context encoder, and a decoder. The contextual information is incorporated into the decoder by attending to the hidden representations from the context encoder with an attention mechanism [22]. For the NAR model, denoted as Fast Contextual Spelling Correction (FCSC), we directly feed the output of text encoder into the decoder. The decoder attends to the context encoder and identifies locations that should be corrected and the candidate context index at each position. One of the advantages is that we can generate results in parallel and don't need to conduct label-by-label prediction in AR, and the decoding speed is also increased. It should be noted that by FCSC we change this problem from a generation task to a classification task. CSC has the potential to correct any error made by the ASR system while FCSC focuses on biasing phrases, which is also a main difference between CSC and FCSC. During inference, filter mechanisms for the context list are proposed to improve inference efficiency and deal with the scalability issues for large context lists. Performance balancing mechanisms are also proposed to adjust biasing degree and control possible WER regressions on general utterances that we do not want to bias. In addition, we demonstrate that the model is a general contextual biasing solution which is effective among different domains. Across several experiments, we find that the proposed method significantly outperforms the baseline contextual FST biasing method, and additional improvements can be achieved by further combining the proposed method on top of FST biasing, leading to the best performance. Additionally, compared to the AR model, the NAR solution

Xiaoqiang Wang, Yanqing Liu, and Sheng Zhao are with Microsoft, China (e-mail: {xiaoqwa, yanqliu, szhao}@microsoft.com).

Jinyu Li, Veljko Miljanic and Hosam Khalil are with Microsoft, US (e-mail: {jinyli, veljkom, hosamk}@microsoft.com).

Manuscript received xxx xxx, xxx; revised xxx xxx, xxx.

reduces model size by 43.2% and speeds up the inference by 2.1 times while achieving WER improvement.

II. Related Work

A. Contextual LM Biasing

A straightforward way to do contextual biasing is combining a contextual LM into the ASR system by shallow fusion, this language model is generally constructed by context phrases as an FST. For the E2E ASR system, it's implemented by interpolating the model posterior probabilities P with the scores P_c from an external contextual LM during beam-search decoding:

$$y^* = \arg\max_y logP(y|x) + \lambda logP_c(y), \tag{1}$$

where λ is a tunable parameter which decides the weight of contextual LM. The contextual LM is constructed by compiling the list of biasing phrases into FST. This method is denoted as FST biasing below. In this research line, several techniques have been explored to improve the biasing performance for the E2E ASR system. [14] proposed to bias the E2E model by applying the contextual LM score boosts at word boundaries. This method cannot deal with proper nouns well because the E2E model uses graphemes or wordpieces during beam search and works at subword unit level. To deal with this problem, [16] proposed to push the weights of the subword FST to each subword unit, which is found to be more effective in E2E ASR biasing. Techniques including adding failure arcs, biasing before beam pruning, biasing at the wordpiece level rather than grapheme level, and adding activation prefix to avoid regression on utterances that do not contain any biasing phrase (anti-context) are also explored to further improve the model performance [16, 17].

B. Bias Encoder

Contextual biasing through shallow fusion is not jointly optimized with the training of the ASR model, which goes against the benefits of direct objective optimization of E2E models. To address this problem, the methods that incorporate contextual information by introducing an additional bias encoder into the E2E framework were proposed, such as CLAS [16], contextual RNN-T [19] and Phoebe [20]. This kind of method trains the E2E ASR model together with a bias encoder which encodes a context list created from the transcripts associated to the utterances in the training batch, and the decoder attends to both the audio encoder and bias encoder to bias output distribution during decoding. In inference, the context phrase list is incorporated as the input of bias encoder. However, this method is more expensive in training and inference [16] and changes the structure of raw ASR models. Some other works, such as trie-based deep biasing [23], two-step memory enhanced model [24], tree-constrained pointer generator component [25] are also investigated to further improve the model performance.

C. Spelling Correction and deliberation

The current research on spelling correction and deliberation approaches mainly try to improve the general ASR accuracy without contextual information. Spelling correction models correct errors in ASR outputs in an E2E manner. In this research line, [26] first proposed an LSTM-based seq-to-seq model which was trained on synthesized speech generated from text-only data. The results showed reasonable improvements compared to simple LM rescoring methods. After that, more structures and training strategies have been proposed, such as a transformer-based model [27] for a Mandarin speech recognition task, models initialized from a pre-trained BERT model [28] or RoBERTa model [3], and so on. Different from spelling correction model, deliberation model [29–31] attends to both first-pass decoding hypothesis and acoustic features to further improve recognition accuracy. One disadvantage of spelling correction or deliberation approaches is that it's hard to do streaming, and there is possible extra model size and latency impact for E2E ASR system due to additional decoding pass. To mitigate the latency issue, [32] proposed an NAR spelling correction model FastCorrect which adds a length predictor to predict the target token number for each source token and generate target sequence input. Though the latency is largely reduced, there is still some regression compared to the AR model.

III. Methodology

A. Autoregressive Contextual Spelling Correction

1) Model structure: As shown in Figure 1, this model is a seq2seq [33] model with a text encoder, a context encoder, and a decoder, which takes the ASR hypothesis as the text encoder input and the context phrase list as the context encoder input. The context encoder encodes each context phrase as hidden states, these hidden states are averaged by context embedding generator to get context embedding. The decoder takes the output of the previous step as input autoregressively, and attends to the outputs of both encoders. These attentions are then added up to generate the final attention, from which the decoder obtains information from ASR hypothesis and context phrase embeddings to correct contextual misspelling errors. All the components are transformer-based [34] and composed by pre-LayerNorm [35, 36], self-attention, encoder-decoder attention and feed-forward layer (FFN). Because the inputs of text encoder and context encoder are both transcriptions, it's natural to share the parameters of these two encoders, which reduces the model size and also helps context encoder training. The final loss is the cross entropy of output probabilities and ground truth label.

B. Non-autoregressive Contextual Spelling Correction

1) Model structure: As shown in Figure 2, similar to AR CSC model, the proposed NAR model (FCSC) contains a text (ASR hypothesis) encoder, a context encoder and a decoder, where the text encoder takes ASR decoding results as input and the context encoder takes biasing phrase list as input. The parameters of the two encoders are shared. The decoder

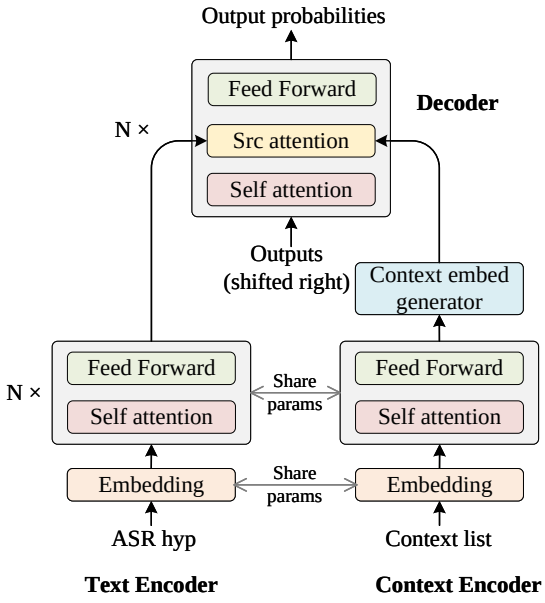


Fig. 1. Autoregressive contextual spelling correction (CSC) model, which contains a text encoder, a context encoder and a decoder, the two encoders share parameters.

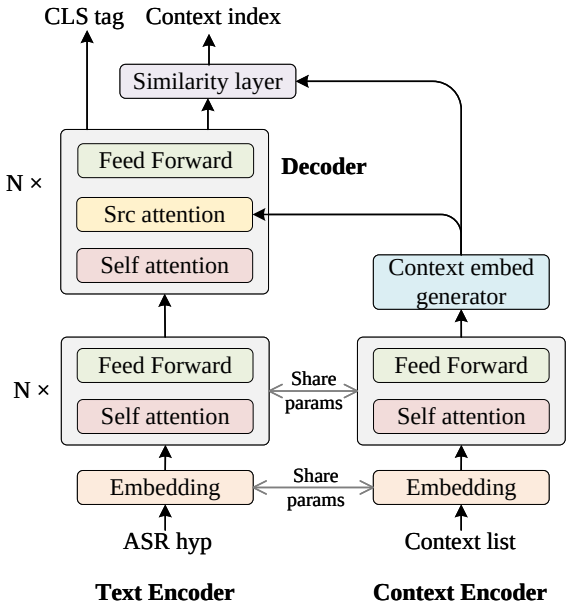


Fig. 2. Non-autoregressive contextual spelling correction (FCSC) model, the decoder directly takes the text encoder output as input, and has two outputs: CLS tag and Context index.

directly takes the output of text encoder as input and attends to the context encoder to decide where to correct and select which context phrase to correct. The encoder and decoder are both transformer-based, the output hidden states of each context phrase are averaged to obtain the context phrase embedding by context embedding generator. The similarity layer calculates the similarity of decoder output hidden states with context embeddings by an inner product operation:

$$s_{ij} = \text{softmax}(\frac{Q_i W^Q (K_j W^K)^T}{\sqrt{d_k}}), \tag{2}$$

where Q_i is the decoder hidden state at i-th position, K_j is the j-th context phrase embedding, and d_k is the dimension of K.

2) Contextual biasing mechanism: The decoder has two outputs:

CLS tag cls. The position-wise classification (CLS) tag cls has the same sequence length as input ASR hypothesis, which determines whether to correct the token at this position or not. It uses "BILO" representation where "B", "I" and "L" represent the beginning, inside and last position of a context phrase, "O" represents a general position outside of a context phrase.

Context index $cind$. Context index is the output of similarity layer, which is the expected index of the ground-truth context phrase in the bias list for this position. We add an empty context at the beginning of the bias list, hence the context index for general tokens that should not be corrected is 0. As shown in Equation 2, the output hidden dimension of similarity layer at each position i is the same as the input bias list size, and the context phrase corresponding to the largest value in s_i is selected during decoding for this position:

$$cind_i = \arg\max s_i. \tag{3}$$

According to the output CLS tag and context index, the final correction results can be determined by replacing the words tagged by CLS tag cls with the context phrase selected by context index $cind$. Here is an input/output example:

biasing phrases c	{"", *"Jack"*, *"Joe Biden"*, ..., *"Tom Jones"*}
ASR hyp x	[*"_who"*, *"_is"*, *"_john"*, *"_b"*, *"ide"*]
CLS tag cls	[O, O, B, I, L]
Context index $cind$	[0, 0, 2, 2, 2]
Final output	[*"_who"*, *"_is"*, *"_joe"*, *"_b"*, *"id"*, *"en"*]

In this example, the context phrase "*Joe Biden*" is recognized as "*John Bide*" by the ASR system, and the index of target context phrase "*Joe Biden*" is 2 in the bias list. By design, the wordpieces of "*John Bide*" should be labeled as [B,I,L] for CLS tag, and the corresponding context index output should be the same to where "*Joe Biden*" is in context list. It should be noted that the final output sequence length can be different from the input due to the replace operation.

The final loss function is the sum of CLS tag loss and context index loss:

$$L = H(\widehat{y_{cls}}, y_{cls}) + H(\widehat{y_{cind}}, y_{cind}). \tag{4}$$

C. Data Processing and Training Strategy

For both CSC and FCSC, we first generate reference-hypotheses pairs from a large set of phrases by using a TTS system. Then the training data is generated during training by combining these context reference-hypotheses pairs with general scripts or sentence patterns. We also use teacher-student learning and quantization to reduce the model size and speed up inference.

1) Text to speech: To prepare the training data, we first collect a large number of context phrases, then a multi-speaker multi-locale (en-*, including en-us, en-gb, en-in, etc.) text

to speech (TTS) system [37] is adopted to generate TTS audio for these phrases. The generated TTS data are then fed into the ASR model to get recognition hypotheses. The input context phrase and the generated hypotheses are paired to get context reference-hypotheses pairs, examples like "*John*"–["*Jane*", "*Jon*", "*June*"]. It should be noted that most context phrases are much shorter than training scripts, which leads to a fast data preparation process both in TTS data generation and ASR inference.

2) Training pairs construction: We also prepare a set of sentence patterns and general scripts to generate training scripts together with the prepared context reference-hypotheses pairs during training. For sentence patterns, we fit the context phrase and one of its hypotheses into the pattern. For general scripts, we randomly replace words with context phrase and its hypothesis. Here is an example:

Context ref-hyp pair	"*John*" – ["*Jane*", "*Jon*", "*June*"]
Pattern	*Call <PersonName> at ten a.m.*
Hyp x	*Call Jon at ten a.m.*
Ref y	*Call John at ten a.m.*
General script	*When do you come to me*
Hyp x	*When do you come to Jane*
Ref y	*When do you come to John*

Where "*Jon*" and "*Jane*" are randomly selected hypothesis of context phrase "*John*" for the sentence pattern and general script, "*me*" is the randomly selected word to be replaced for the general script.

To take care of utterances that do not contain any context phrase but with a biasing phrase list (anti-context) and cases that the target context phrase somehow doesn't appear in the biasing phrase list, we also leave part of the input general scripts in the training set unchanged with probability P_{cont}.

3) Context setup: During training, we randomly sample N_c biasing phrases for each utterance from the whole context list which is pooled from biasing phrases of all utterances in this batch. N_c is randomly sampled from uniform distribution $[1, N_{cmax}]$, where N_{cmax} is the max context list size defined as a training hyper-parameter. To distinguish the sampled context list for each utterance in the batch, a context mask is adopted on top of the context embedding generator. What's more, to increase the diversity of possible error patterns, besides using multi-speaker multi-locale TTS system to generate TTS data, we also swap the reference context phrase and its hypothesis randomly with probability P_m to extend the set of context reference-hypotheses pairs.

4) Teacher-student learning: Teacher-student learning [38, 39] is an effective way to reduce model size and improve inference efficiency. For both CSC and FCSC, we use teacher-student learning to make it smaller and more efficient. The loss function to train the student model contains a hard loss L_{hard} and a soft loss L_{soft}. The hard loss is the cross-entropy of student model output y_S and reference y, and the soft loss is the KL-divergence of y_S and teacher model output y_T:

$$L = \alpha L_{soft} + (1-\alpha) L_{hard} \tag{5}$$

$$L_{hard} = H(y_S, y) \tag{6}$$

$$L_{soft} = D_{\mathrm{KL}}\left(\mathrm{softmax}(\frac{y_S}{T}), \mathrm{softmax}(\frac{y_T}{T})\right) \cdot T^2 \tag{7}$$

where T is the temperature to adjust the smoothness of output probabilities, α determines the proportion of hard loss and soft loss. For FCSC, the final hard/soft loss is the sum of the loss of CLS tag cls and Context index $cind$:

$$L_{hard} = L_{hard}^{cls} + L_{hard}^{cind} \tag{8}$$

$$L_{soft} = L_{soft}^{cls} + L_{soft}^{cind} \tag{9}$$

D. Inference

1) Context pre-selection mechanism: For context phrase list, we propose a relevance ranker (rRanker) and a preference ranker (pRanker) to preselect context phrases from the raw context list size and deal with the possible scalability issue. The preference ranker represents the preliminary knowledge of the context phrases in terms of preference weight, examples like the call frequency of context phrases in the target domain. The relevance ranker aims to measure the relevance between the specific ASR hypothesis and the given context phrases. To simplify the inference process and control latency, we propose an edit distance-based method to obtain the weight of relevance ranker:

$$W_r^j = -\frac{\min_i(\mathrm{edit\_distance}(c_j, e_i))}{\mathrm{len}(c_j)}, \tag{10}$$

where e_i is the segment cut off from input ASR hypothesis with the same length of the context phrase c_j from the i-th word. The final relevance ranker weight is the minimum value of these edit distance normalized by the length of c_j. For edge cases where the length of remaining characters from the i-th word is shorter than c_j, we simply use the remaining characters as e_i. Here is an example:

e_1 e_3 e_6
Please send a message to Ernest
c_j: Earnest

where e_1 and e_3 are the first and third segments cut off from the first and third word with the same length of c_j. e_6 is an edge case which is shorter than c_j. The minimum edit distance should be obtained at e_6. After that, the preference ranker weight W_p and relevance ranker weight W_r are combined with weight α_p and $1-\alpha_p$, respectively. The top K_f context phrases are then selected out from the raw K_r bias phrases as the final input for context encoder:

$$c_1, c_2, ..., c_{K_f} = \arg\operatorname*{topK}_j(\alpha_p W_p^j + (1-\alpha_p) W_r^j). \tag{11}$$

2) FCSC output format: For FCSC, the final output is determined based on the model outputs CLS tag cls and context index $cind$. However, during inference, the output format is not always as standard as what training set looks like, which is referred to as illegal output. Some examples are listed in Table I. This happens when the model is not that confident of the output, so we will give up correcting such cases.

TABLE I
OUTPUT EXMAPLES OF cls AND $cind$ FOR FCSC

Outputs	Comments	Legal or not
cls: [O, O, B, I, L, O] $cind$: $[0, 0, 9, 9, 9, 0]$	correct	✓
cls: [O, O, B, I, I, O] $cind$: $[0, 0, 6, 6, 6, 0]$	incomplete cls where the end position tag L is lost	✗
cls: [O, O, B, L, O] $cind$: $[0, 4, 4, 0, 0]$	cls and $cind$ inconsistency	✗
cls: [O, B, I, L] $cind$: $[0, 4, 5, 4]$	tagged position corresponds to multiple context indexes	✗

3) Performance balancing: When the model is too "biased", the model may suffer from regressions on anti-context cases. FST biasing generally uses an interpolation weight λ to balance the model performance among biasing sets and anti-context cases. For the proposed method, a part of training set are general scripts without context phrase, which teaches the model to decide when to correct according to the given context phrase list and the input ASR hypothesis, hence the model can effectively control the regression on anti-context cases by itself. However, regressions still exist for some cases and there may be additional requirements on the regression in real application. Here we propose to use NER detector and controllable parameter s^o to control the regression of CSC/FCSC. We will show the effectiveness of these methods in the experiment section below.

NER detector.

The first solution is adopting an NER detector before CSC/FCSC inference, which classifies whether the ASR hypothesis contains terms to be biased and decides when to activate CSC/FCSC. In general, this detector can be designed as a classification model, specifically, a Named Entity Recognition (NER) task [40, 41] which classifies entities in the given ASR hypothesis. The training data for this classification model can be obtained following the methods in NER tasks. For narrow domains with representative sentence patterns, rule-based detector is enough to achieve reasonable classification precision and recall. The "bias prefixes" used by [16, 17] is indeed a subset of rule-based detector. However, just prefix information is limited, especially for cases that without prefix (e.g., "*Harry is my good friend*") and cases that with too general prefix (e.g., "*schedule a meeting for me and Jessa on Tuesday*" where "*and*" is the prefix). For the proposed model, we can use prefixes, suffixes, and sentence patterns to make a more reliable classification because we can see the full ASR hypothesis. In our experiments, some of the baseline ASR models are able to tag out entities with classification tokens such as <*name*> and <*entity*>, which can be directly adopted as NER detector, denoted as **token filter (tfilter)**. For baseline ASR models without such tokens, a rule-based detector is used for convenience, denoted as **pattern filter (pfilter)**. What's more, for multiple output candidates of an ASR system, CSC/FCSC is activated as long as one of these candidates passes through the detector.

Controllable threshold s^o.

Although the NER detector has effective regression control ability for anti-context cases, it's still time consuming and not easy to extract the prefixes/suffixes/patterns from an outside adaptation set, there are also many things to consider to balance the classification precision and recall. Therefore, for FCSC, we also propose another mechanism to easily control the regression by just adjusting a controllable threshold parameter s^o, which measures the confidence of the model output by adopting the output probabilities s_{ij} from similarity layer:

$$s_i^* = \underset{i}{\text{mean}}(\max_j(s_{ij})), \quad i \in [i_s, i_e] \tag{12}$$

$$cind_i = \begin{cases} \arg\max s_i & s_i^* >= s^o \\ 0 & s_i^* < s^o \end{cases} \tag{13}$$

Where $[i_s, i_e]$ is a continuous sequence labeled out by CLS tag, for example, for output $\{cls\text{: } [O, O, B, I, L, O]\}$, i_s is 2 and i_e is 4. s_i^* represents the confidence of the model for the current output, and s^o is a threshold value which ranges from 0 to 1. This threshold can be tuned for each application scenario to meet the regression requirements. When $s_i^* < s^o$, this position is left unchanged due to low confidence.

4) Score interpolation: ASR system may output multiple candidates for each utterance, these candidates contain more signals and help improve the model performance according to our experiments. However, too many input candidates will increase the latency, we select top N_{asr} candidates for CSC/FCSC decoding, where N_{asr} is determined by the model performance and latency requirements.

CSC conducts beam search process during decoding, which generates the corresponding CSC hypotheses $\{H_{i1}, H_{i2}, ..., H_{iN_{csc}}\}$ for each ASR hypothesis H_i. The final decoding results are obtained by ranking these $N_{asr} \times N_{csc}$ hypotheses:

$$H^* = \arg\max_H \lambda_{ASR} logp_i + \lambda_{CSC} logp'_{ij} \tag{14}$$

where λ_{ASR} and λ_{CSC} are the weights for ASR and CSC scores. p_i is the utterance level probability of H_i, and p'_{ij} is the utterance level probability of the j-th CSC hypothesis for H_i.

Unlike the AR method, there is no beam search process for FCSC. The most possible hypothesis is obtained by the following criteria:

$$H^* = \arg\max_H \lambda_{ASR} logp_i + \lambda_{FCSC} logq_i \tag{15}$$

where $logq_i$ is the utterance level FCSC score which is the sum of utterance level log probabilities of cls and $cind$:

$$logq_i = logq_i^{cls} + logq_i^{cind} \tag{16}$$

IV. EXPERIMENT

A. Data Sets

The training context list includes 1M names generated from 48k name words, and 6M proper nouns collected from two open source datasets NELL [42] and Yago [43]. We use a multi-speaker multi-locale(en-*) TTS [37] AM which contains

TABLE II
TEST SETS

Test set	#utt.	context type	context list size	context examples
Name set	11597	person name	1509	Jotham Parker
Personal assistant (PA) dev	830	person name	1507	Andrew
Personal assistant (PA) blind	1024	person name	1507	Xiaofang
Random8k set	8000	-	-	-
Text editor	797	commands	210	Uppercase the next paragraph
Medical set	516	medical health related	11638	Pause cardiac output monitoring

363 speakers to get TTS audio, the generated TTS data are then fed into an RNN-T model to obtain ASR hypotheses. A context phrase dictionary is then constructed from these reference-hypotheses pairs, in which each reference context phrase corresponds to a list of hypotheses. What's more, 512 sentence patterns and 26M in-house general scripts are used to construct training set with the generated context phrase dictionary as the method mentioned before. We have open-sourced the TTS synthetic data for the collected entities at https://pan.baidu.com/s/1UajBs6zlzE2P9k0cqJQxSg?pwd=9d7m.

We evaluate model performance on several test sets, detailed in Table II. The Random8k set contains anti-context utterances, the context list of this set is randomly selected from Name set, which aims to ensure that the model doesn't affect recognition quality if all biasing phrases are irrelevant. To evaluate the model performance on general context biasing in other domains, there are also two test sets in which biasing phrases are general phrases rather than person names or proper nouns, the biasing phrases of these sets are frequently used terms in its own domain but not related to training set.

B. Experimental Setup

During training, we set the batch size to 300, the max size of context phrase list N_{cmax} of each utterance to 100, the ratio of general utterances without context phrase P_{cont} to 20%, and the ratio of swapping context reference-hypotheses pairs P_m to 20%. We use teacher-student learning and quantization to reduce model size. The student model was trained with the same training set as teacher, and we set T to 1 and α to 0.9. The model parameters of teacher and student are listed in Table III.

During inference, we set the size of filtered context list K_f to 100 to be consistent with training. We decode with two baseline ASR models: an RNN-T model and a T-T model, trained with 64 thousand hours Microsoft anonymized data. The RNN-T model is able to output <*name*> tokens which classifies person names (tfilter), which can be directly used as NER detector for both CSC and FCSC. A pattern filter (pfilter) which is generated from an outside adaptation set is adopted for T-T model. Similarly, the baseline FST biasing method uses <*name*> tokens for RNN-T and prefix set extracted from the same adaptation set for T-T as activation prefix. For RNN-T, top 4 candidates are selected to correct, while for T-T, we select top 1 because the T-T model we use just supports single candidate output. We also test FCSC model performance when with the proposed controllable threshold s^o rather than NER detector.

TABLE III
MODEL PARAMETERS

Model		layers	d_model	heads	d_FFN	Params(M)
CSC	Teacher	6	512	8	2048	55.4
	Student	3	192	4	768	5.2
FCSC	Teacher	6	512	8	2048	47.4
	Student	3	192	4	768	4.2

C. Performance

1) WER reduction: The results on the first three test sets in name domain and the Random8k set are listed in Table IV, where "Bias" represents the baseline FST biasing method. Compared to baseline ASR models, the proposed model achieves as much as 41.7% relative word error rate reduction on RNN-T and 51.0% on T-T. Compared to the FST biasing method, the proposed model achieves comparable or much better performance among all these sets. We also test the model performance based on RNN-T+Bias and T-T+Bias, and find our method can still achieve significant gain based on FST biasing as much as 32.6% on RNN-T and 38.5% on T-T, which indicates that the proposed model can work on top of the FST biasing method to achieve the best performance for E2E ASR customization. Compared to AR CSC model, the NAR solution still performs better among all the test sets, which is slightly different from other NAR solutions. The reason, we believe, is that different from other NAR architectures, our method doesn't need to align the encoder and decoder by adopting a length or duration predictor, which avoids possible error accumulation, the structure we use is also very suitable for this task.

Table V lists the results on the two general biasing sets. Compared to baseline RNN-T, the model achieves as much as 50.7% relative WER improvement. When correct on top of FST biasing, 24.6% relative WER improvement can also be achieved. These results demonstrate that the proposed method is a general contextual biasing solution which is domain-insensitive and is effective among different scenarios.

2) Latency improvement: The model size and latency of the proposed AR CSC model and the NAR FCSC model are listed in Table VI. The latency is the decoding time of CSC/FCSC model per utterance in the test set regardless of baseline ASR model, which is tested on a machine with 2.60GHz CPU using single thread, each utterance corresponds to 4 ASR output candidates. The NAR model reduces model size by 43.2% and speeds up the inference by 2.1 times, which shows huge advantages for the NAR model to be deployed on

TABLE IV
MODEL PERFORMANCE ON NAME DOMAIN

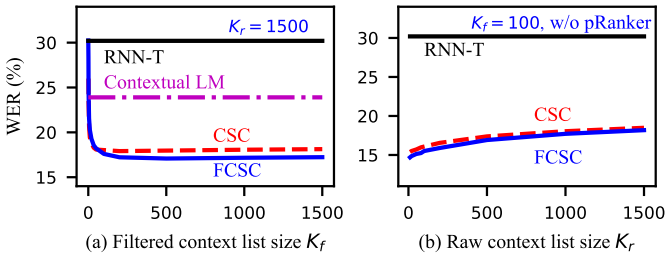
Method	Name set		PA dev		PA blind		Random8k set	
	WER	WERR	WER	WERR	WER	WERR	WER	WERR
RNN-T	30.2	-	24.2	-	22.5	-	13.9	-
+CSC	18.1	40.1	17.4	28.1	15.3	32.0	14.4	−3.6
+FCSC	17.6	41.7	16.8	30.6	14.6	35.1	14.4	−3.6
+FCSC (s^o)	18.0	40.4	17.3	28.5	15.2	32.4	14.0	−0.7
RNN-T+Bias	23.9	20.8	19.5	19.4	17.1	24.0	14.0	−0.7
+CSC	16.9	44.0	16.7	31.0	14.1	37.3	14.6	−5.0
+FCSC	16.1	46.7	15.5	36.0	13.1	41.8	14.4	−3.6
+FCSC (s^o)	16.3	46.0	15.9	34.3	13.1	41.8	14.1	−1.4
T-T	24.1	-	17.5	-	16.6	-	8.8	-
+FCSC	11.8	51.0	12.7	27.4	11.2	32.5	8.8	−0.0
+FCSC (s^o)	11.9	50.6	11.9	32.0	11.3	31.9	9.0	−2.3
T-T+Bias	17.4	27.9	12.6	27.9	12.6	24.0	8.9	−1.1
+FCSC	10.7	55.6	10.4	40.6	10.3	38.0	9.0	−2.3
+FCSC (s^o)	10.7	55.6	9.9	43.4	10.2	38.6	9.0	−2.3

TABLE V
MODEL PERFORMANCE ON GENERAL BIASING SETS

Method	Text editor		Medical set	
	WER	WERR	WER	WERR
RNN-T	14.6	-	14.0	-
+FCSC	7.2	50.7	8.6	38.6
RNN-T+Bias	10.3	29.3	6.9	50.8
+FCSC	7.9	45.9	5.2	62.9

TABLE VI
MODEL SIZE AND LATENCY

	model	teacher	student
size(MB)	CSC	223.0	9.5
	FCSC	212.0	5.4
latency(ms/utt)	CSC	793.6	132.3
	FCSC	530.0	63.7



(a) Filtered context list size K_f (b) Raw context list size K_r

Fig. 3. Influence of filtered context list size K_f and raw context list size K_r on moder performance. WER increases with Kr in a concave curve shape, and FCSC achieves stable improvement over CSC.

device compared to the AR solution. It should be noted that this latency includes context phrase encoding, which can be calculated in advance and loaded as cache in real application, therefore, the latency can be further reduced and optimized in runtime.

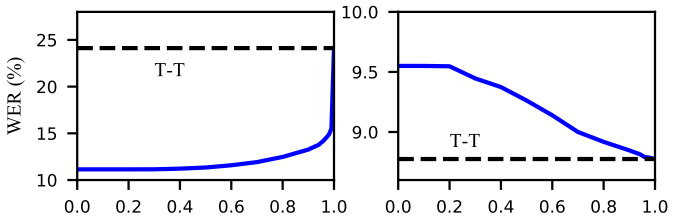
3) Influence of context list size: Figure 3 illustrates the influence of filtered context list size K_f and raw context list size K_r on Name set. Where K_r depends on the given context phrase list, and K_f is a decoding parameter that can be directly adjusted during inference. Here K_r is adjusted by randomly sampling a subset of context phrases from the raw context phrase list while keeping the target phrase in the set. We can see WER increases with K_r in a concave curve shape for both CSC and FCSC, which means raw context list size does not influence the model performance much when it's large enough, the proposed method can deal with the scalability problem well. The figure also shows the WER curve of FCSC lies below CSC in most of time for both K_r and K_f, which indicates FCSC achieves stable improvement over CSC with decoding parameter change.

4) Performance balancing of FCSC: As discussed before, for FCSC, we propose a regression control mechanism which adopts controllable threshold s^o to balance model performance between biasing set and anti-context cases. The model performance with $s^o = 0.7$ on name domain is listed in Table IV, which shows similar performance with the method that adds NER detector. Figure 4 illustrates the effect of s^o to model performance on Name set and Random8k set in details. $s^o = 0$ is the vanilla setting which means we don't use this mechanism while $s^o = 1$ means we totally give up correction. With the increase of s^o, the WER increases on Name set while decreases quickly on Random8k set, which demonstrates the effectiveness of the proposed balance mechanism. We define r as the relative WER gap narrowing ratio:

$$r = \frac{W - W_0}{W_1 - W_0}, \tag{17}$$

where W represents WER, W_0 and W_1 represent the WER when $s^o = 0$ and $s^o = 1$. The variation of r with the change of s^o is illustrated in Figure 5. We can see the WER on Name set experiences a long flat period when s^o is small and then encounters steep increase only when near to 1.0, which means most of the cases in Name set are corrected with enough confidence and guarantees enough safe space for us to conduct performance balancing.

5) OOV performance: Table VII lists model performance of out-of-vocabulary (OOV) terms and in-vocabulary (IV)



(a) Effect of s^o on Name set (b) Effect of s^o on Random8k set

Fig. 4. Control model performance among different test sets with threshold s^o. With the increase of s^o, the WER increases slowly on Name set while decreases quickly on Random8k set (which contains anti-context cases).

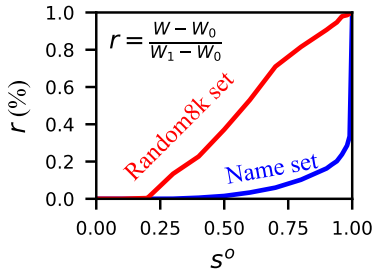


Fig. 5. WER relative change with s^o. The WER on Name set experiences a long flat period and then encounters steep increase only when near to 1.0, which guarantees enough safe space to conduct performance balancing.

terms on different test sets, where OOV rate represents the proportion of utterances with OOV biasing phrases not seen in CSC/FCSC model training. It should be noted that most of the bias phrases in Text editor set and Medical set are not seen in training set. The results show good performance on OOV terms, which indicates that the model learns error patterns in wordpiece level rather than word level. We also find that the performance on IV terms is generally worse, we checked the results and the explanation is that for the three sets with names, the raw WER of IV terms is relatively low and contains less error for correction; for the Text editor set and Medical set, there are only few samples in IV set and not that representative. In general, OOV terms will easily appear in domains which are not related to training, the OOV performance also demonstrates the general contextual biasing ability of the proposed method from another point of view.

TABLE VII
OOV PERFORMANCE (WER)

dataset	OOV rate	model	OOV	IV
Name set	54.0%	T-T	32.7	16.8
		+FCSC	15.3	11.3
PA dev	42.4%	T-T	24.5	13.3
		+FCSC	16.2	11.0
PA blind	32.3%	T-T	26.8	11.1
		+FCSC	15.6	9.5
Text editor	82.6%	RNN-T	14.7	14.1
		+FCSC	6.4	11.6
Medical set	93.8%	RNN-T	14.1	10.7
		+FCSC	8.6	8.9

V. CONCLUSIONS

In this paper, we introduce a novel contextual biasing method for customizing end-to-end ASR systems. Our method integrates a context encoder into the spelling correction model with carefully designed AR and NAR mechanisms. Novel filtering algorithms are designed to deal with the large size context list. Effective performance balancing mechanisms are proposed to balance model performance on anti-context terms. The method is also a general contextual biasing solution which is domain-insensitive and can be adopted in different scenarios. Empirical studies have demonstrated that the proposed method significantly improves the ASR model performance on contextual biasing tasks, which also shows competitive or better performance over conventional methods. The NAR architecture achieves smaller model size, lower latency, and better performance than the AR design, which is more competitive for on-device application. However, there are still some limitations of the proposed method, including not being acoustic-grounded and the extra complexity in being streaming for the NAR solution. In future work, we would like to explore integrating additional acoustic information into the model to further improve the performance. We would also explore methods to enable streaming for the NAR model to further reduce latency.

REFERENCES

[1] J. Li, "Recent advances in end-to-end automatic speech recognition," *APSIPA Transactions on Signal and Information Processing*, vol. 11, no. 1, 2022.

[2] T. N. Sainath, Y. He, B. Li, A. Narayanan, R. Pang, A. Bruguier, S.-y. Chang, W. Li, R. Alvarez, Z. Chen *et al.*, "A streaming on-device end-to-end model surpassing server-side conventional model quality and latency," in *Proc. ICASSP*. IEEE, 2020, pp. 6059–6063.

[3] J. Li, R. Zhao, Z. Meng, Y. Liu, W. Wei, S. Parthasarathy, V. Mazalov, Z. Wang, L. He, S. Zhao, and Y. Gong, "Developing RNN-T models surpassing high-performance hybrid models with customization capability," in *Proc. Interspeech*, 2020.

[4] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[5] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[6] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proc. ICASSP*. IEEE, 2016, pp. 4960–4964.

[7] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, and R. Pang, "Streaming end-to-end speech recognition for mobile devices," in *Proc. ICASSP*, 2019.

[8] J. Li, R. Zhao, H. Hu, and Y. Gong, "Improving RNN transducer modeling for end-to-end speech recognition," in *Proc. ASRU*, 2019.

[9] G. Saon, Z. Tüske, and K. Audhkhasi, "Alignment-length synchronous decoding for RNN transducer," in *Proc. ICASSP*, 2020, pp. 7804–7808.

[10] A. Zeyer, A. Merboldt, R. Schlüter, and H. Ney, "A new training pipeline for an improved neural transducer," in *Proc. Interspeech*, 2020.

[11] C.-F. Yeh, J. Mahadeokar, K. Kalgaonkar, Y. Wang, D. Le, M. Jain, K. Schubert, C. Fuegen, and M. L. Seltzer, "Transformer-transducer: End-to-end speech recognition with self-attention," *arXiv preprint arXiv:1910.12977*, 2019.

[12] Q. Zhang, H. Lu, H. Sak, A. Tripathi, E. McDermott, S. Koo, and S. Kumar, "Transformer transducer: A streamable speech recognition model with transformer encoders and RNN-T loss," in *Proc. ICASSP*, 2020, pp. 7829–7833.

[13] X. Chen, Y. Wu, Z. Wang, S. Liu, and J. Li, "Developing real-time streaming transformer transducer for speech recognition on large-scale dataset," *arXiv preprint arXiv:2010.11395*, 2020.

[14] I. Williams, A. Kannan, P. Aleksic, D. Rybach, and T. Sainath, "Contextual speech recognition in end-to-end neural network sys-tems using beam search," in *Proc. Interspeech*, 2018.

[15] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang *et al.*, "Streaming end-to-end speech recognition for mobile devices," in *Proc. ICASSP*, 2019, pp. 6381–6385.

[16] G. Pundak, T. N. Sainath, R. Prabhavalkar, A. Kannan, and D. Zhao, "Deep context: end-to-end contextual speech recognition," in *Proc. SLT*, 2018, pp. 418–425.

[17] D. Zhao, T. N. Sainath, D. Rybach, D. Bhatia, B. Li, and R. Pang, "Shallow-fusion end-to-end contextual biasing," in *Proc. Interspeech*, 2019.

[18] D. Le, G. Keren, J. Chan, J. Mahadeokar, C. Fuegen, and M. L. Seltzer, "Deep shallow fusion for RNN-T personalization," in *Proc. SLT*, 2020.

[19] M. Jain, G. Keren, J. Mahadeokar, G. Zweig, F. Metze, and Y. Saraf, "Contextual RNN-T for open domain ASR," in *Proc. Interspeech*, 2020.

[20] A. Bruguier, R. Prabhavalkar, G. Pundak, and T. N. Sainath, "Phoebe: Pronunciation-aware contextualization for end-to-end speech recognition," in *Proc. ICASSP*, 2019, pp. 6171–6175.

[21] X. Wang, Y. Liu, S. Zhao, and J. Li, "A light-weight contextual spelling correction model for customizing transducer-based speech recognition systems," in *Proc. Interspeech*, 2021.

[22] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[23] D. Le, M. Jain, G. Keren, S. Kim, Y. Shi, J. Mahadeokar, J. Chan, Y. Shangguan, C. Fuegen, O. Kalinli, Y. Saraf, and M. L. Seltzer, "Contextualized streaming end-to-end speech recognition with trie-based deep biasing and shallow fusion," in *Proc. Interspeech*, 2021.

[24] C. Huber, J. Hussain, S. Stüker, and A. Waibel, "Instant one-shot word-learning for context-specific neural sequence-to-sequence speech recognition," *arXiv preprint arXiv:2107.02268*, 2021.

[25] G. Sun, C. Zhang, and P. C. Woodland, "Tree-constrained pointer generator for end-to-end contextual speech recognition," *arXiv preprint arXiv:2109.00627*, 2021.

[26] J. Guo, T. N. Sainath, and R. J. Weiss, "A spelling correction model for end-to-end speech recognition," in *Proc. ICASSP*, 2019, pp. 5651–5655.

[27] S. Zhang, M. Lei, and Z. Yan, "Investigation of transformer based spelling correction model for CTC-based end-to-end Mandarin speech recognition," in *Proc. Interspeech*, 2019.

[28] O. Hrinchuk, M. Popova, and B. Ginsburg, "Correction of automatic speech recognition with transformer sequence-to-sequence model," in *Proc. ICASSP*, 2020, pp. 7074–7078.

[29] K. Hu, T. N. Sainath, R. Pang, and R. Prabhavalkar, "Deliberation model based two-pass end-to-end speech recognition," *arXiv preprint arXiv:2003.07962*, 2020.

[30] G. Ye, V. Mazalov, J. Li, and Y. Gong, "Have best of both worlds: two-pass hybrid and e2e cascading framework for speech recognition," in *Proc. ICASSP*, 2022.

[31] W. Wang, K. Hu, and T. Sainath, "Deliberation of streaming rnn-transducer by non-autoregressive decoding," *arXiv preprint arXiv:2112.11442*, 2021.

[32] Y. Leng, X. Tan, L. Zhu, J. Xu, R. Luo, L. Liu, T. Qin, X.-Y. Li, E. Lin, and T.-Y. Liu, "Fastcorrect: Fast error correction with edit alignment for automatic speech recognition," in *Proc. NeurIPS*, 2021.

[33] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *arXiv preprint arXiv:1409.3215*, 2014.

[34] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 6000–6010.

[35] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[36] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI blog 1.8*, 2019.

[37] Y. Liu, Z. Xu, G. Wang, K. Chen, B. Li, X. Tan, J. Li, L. He, and S. Zhao, "Delightfultts: The microsoft speech synthesis system for blizzard challenge 2021," *arXiv preprint arXiv:2110.12612*, 2021.

[38] J. Li, R. Zhao, J.-T. Huang, and Y. Gong, "Learning small-size DNN with output-distribution-based criteria." in *Proc. Interspeech*, 2014, pp. 1910–1914.

[39] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[40] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification," *Lingvisticae Investigationes: 30.1*, pp. 3–26, 2007.

[41] J. Li, A. Sun, J. Han, and C. Li, "A survey on deep learning for named entity recognition," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[42] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling, "Never-ending learning," in *Proc. AAAI*, 2015.

[43] T. P. Tanon, G. Weikum, and F. Suchanek, "Yago 4: A reason-able knowledge base," in *Extended Semantic Web Conference*, 2020.