Harnessing Instruction-Tuned Large Language Model for Guiding End-to-End Speech Recognition

Anonymous ACL submission

Abstract

Modern large language models (LLMs) are adept at performing various text generation tasks when prompted with instructions designed for specific objectives. These abilities can enhance the quality of text produced by automatic speech recognition (ASR), enabling the selection of words that are more semantically accurate. However, relying solely on LLMs to correct errors in ASR predictions may lead to unintended word generations or modifications that do not accurately reflect the speech input. In this work, we propose a novel ASR model that integrates the text generation capabilities of LLMs, while ensuring proper alignment with speech inputs. Specifically, our model is built on the attention-based encoder-decoder (AED) structure, with the LLM serving as a front-end feature extractor for the decoder. The decoder is trained to predict words from the LLM-derived features, where cross-attention accounts for aligning these features with the speech encodings from the encoder. We also design an effective prompting strategy that uses a hypothesized text sequence to extract linguistic information beneficial for performing ASR. Experimental results demonstrate that our proposed model outperforms conventional AED-based models across major ASR tasks.

1 Introduction

In the field of natural language processing (NLP), the pre-training of language models (LMs) has become a dominant paradigm. This process involves training LMs on vast amounts of text data using self-supervised objectives (Devlin et al., 2019; Radford et al., 2018), enabling the acquisition of versatile linguistic representations that enhance the performance across various downstream tasks (Wang et al., 2018; Gao et al., 2021). In light of the remarkable success in NLP, pre-trained LMs have increasingly been adopted for speech processing tasks. Particularly in end-to-end automatic speech recognition (ASR), the linguistic knowledge from pre-trained LMs has proven beneficial in generating accurate textual outputs (Salazar et al., 2020; Futami et al., 2020; Yi et al., 2021; Zheng et al., 2021; Deng et al., 2022; Higuchi et al., 2022), providing semantic and morphosyntax information (Tenney et al., 2019) — often challenging to capture in end-to-end ASR training with limited transcription data.

Recent focus has centered on the use of rapidly advancing pre-trained *large* LMs (LLMs) (Radford et al., 2019; Brown et al., 2020; Scao et al., 2022; Wei et al., 2022b; Touvron et al., 2023a; Chowdhery et al., 2023; OpenAI, 2023), which have demonstrated exceptional versatility in performing diverse text generation tasks with little or even no task-specific training data. LLMs have shown promising results in improving end-to-end ASR performance when used in traditional LM-based decoding methods, such as shallow fusion (Hu et al., 2023) and rescoring (Udagawa et al., 2022; Chen et al., 2023b; Ma et al., 2023a,b; Yang et al., 2023). To fully utilize the inherent capabilities of LLMs, numerous studies have explored effective strategies for directly adapting them to process speech inputs (Wang et al., 2023; Zhang et al., 2023; Chen et al., 2023a; Wu et al., 2023b; Rubenstein et al., 2023; Deshmukh et al., 2023; Nachmani et al., 2024; Yu et al., 2024; Fathullah et al., 2024). Nonetheless, this adaptation often requires fine-tuning the LLMs, which can be computationally expensive and typically requires an additional mechanism to condense speech inputs into a more manageable length. A more straightforward approach is to prompt LLMs to correct grammatical errors (Wu et al., 2023a; Fang et al., 2023) in ASR hypotheses, but the absence of speech information can lead to hallucinations or overcorrections, generating words not present in the speech input.

In this work, we present a novel end-to-end ASR model that efficiently utilizes an LLM to achieve

accurate text generation. The proposed model is based on the attention-based encoder-decoder (AED) architecture, constructed using the joint connectionist temporal classification (CTC) and attention framework (Watanabe et al., 2017). The core component of our model is the LLM-guided decoder, which augments the original decoder by employing a fixed-parameter LLM to serve as a front-end feature extractor. This integration allows the decoder to directly leverage the powerful text generation capabilities of the LLM. Additionally, the cross-attention mechanism facilitates the alignment of the LLM-derived features with the speech information embedded by the encoder. To optimize the extraction of linguistic features beneficial for the decoder, we also design an effective prompting strategy for the LLM, using a hypothesized text sequence generated through CTC decoding.

2 Background

This section outlines the key model formulations essential for understanding the proposed integration of LLMs into end-to-end ASR. First, we discuss an instruction-tuned LLM, emphasizing its prompt-based controllability. Subsequently, we describe an end-to-end ASR model based on the AED architecture and its combination with CTC.

2.1 Instruction-Tuned LLM

The recent LLMs possess the capability to be "prompted" to execute specific tasks. This involves providing instructions or contexts that influence the subsequent output generated by the model, allowing users to flexibly control the model's behavior depending on the need. Additionally, advancements in instruction fine-tuning have further enhanced the LLMs' potential for performing zero-shot task transfer, which helps to produce more precise responses without requiring task-specific retraining or fine-tuning (Wei et al., 2022a; Ouyang et al., 2022; Chung et al., 2024).

We focus on Llama2-Chat, an instruction-tuned version of Llama2 (Touvron et al., 2023b), as a pre-trained LLM used in this work. We hereafter refer to this chat model as "Llama2" for brevity. Llama2, consisting of deep Transformer decoder-based layers (Vaswani et al., 2017; Radford et al., 2018), outputs a D^{llm}-dimensional hidden vector $\mathbf{e}_n$ at a token position n as

$$\mathbf{e}_n = \text{Llama2}(\underbrace{W^{\mathsf{ins}}}_{\text{Instruction}}, \underbrace{W^{\mathsf{usr}}}_{\text{User Input}}, \underbrace{W_{<n}}_{\text{Response}}), \quad (1)$$

where $W^{\mathsf{ins}} \in \mathcal{V}^{N^{\mathsf{ins}}}$ is an N^{ins}-length instruction sequence that specifies the details of a task; $W^{\mathsf{usr}} \in \mathcal{V}^{N^{\mathsf{usr}}}$ is an N^{usr}-length user input sequence that serves as the given input for the task; $W = (w_n \in \mathcal{V} | n = 1, \cdots, N)$ is an N-length response sequence generated by the LLM; and $\mathcal{V}$ is the vocabulary of Llama2. The previous tokens are represented as $W_{<n} = (w_0, \cdots, w_{n-1})$, where $w_0 =$ <s> is a start-of-sentence symbol. Typically, the prefix, combining W^{ins} and W^{usr}, is referred to as a **prompt**, which guides the model to generate responses W in a specific manner. See the Llama2 input in Fig. 1 for example input sequences.

Llama2 computes the likelihood of a target sequence W as

$$p(W|W^{\mathsf{ins}}, W^{\mathsf{usr}}) = \prod_{n=1}^{N+1} p(w_n|W_{<n}, W^{\mathsf{ins}}, W^{\mathsf{usr}}), \quad (2)$$

where $w_{N+1} =$ </s> is an end-of-sentence symbol. The probability of generating w_n in Eq. (2) is computed using the output $\mathbf{e}_n$ from Eq. (1) as

$$p(w_n|W_{<n}, W^{\mathsf{ins}}, W^{\mathsf{usr}}) = \sigma(\text{Lin}_{D^{\mathsf{llm}} \to |\mathcal{V}|}(\mathbf{e}_n)), \quad (3)$$

where $\text{Lin}_{D^{\mathsf{llm}} \to |\mathcal{V}|}(\cdot)$ projects a D^{llm}-dimensional feature vector to a logit, and $\sigma(\cdot)$ represents the softmax function.

2.2 Joint CTC/Attention End-to-End ASR

Let $O \in \mathbb{R}^{T \times F}$ denote a T-length input speech sequence with F-dimensional acoustic features and $W \in \mathcal{V}^N$ represent the corresponding target sequence[1]. End-to-end ASR aims to directly map O to W by modeling the posterior distribution of $p(W|O)$ using a single deep neural network.

AED (Chorowski et al., 2015; Chan et al., 2016) formulates end-to-end ASR using a probabilistic chain rule as

$$p^{\mathsf{aed}}(W|O) \triangleq \prod_{n=1}^{N+1} p(w_n|W_{<n}, O). \quad (4)$$

The token emission probability in Eq. (4) is computed as

$$H = (\mathbf{h}_1, \cdots, \mathbf{h}_{T'}) = \text{Encoder}(O), \quad (5)$$

$$p(w_n|W_{<n}, O) = \text{Decoder}(W_{<n}, H), \quad (6)$$

where $\text{Encoder}(\cdot)$ first down-samples O (i.e., $T' = T/4$) and then converts it into a sequence of D^{asr}-dimensional hidden vectors H. $\text{Decoder}(\cdot)$ repre-

[1]Throughout this work, we consistently use the common vocabulary $\mathcal{V}$ of Llama2 for tokenizing text sequences.

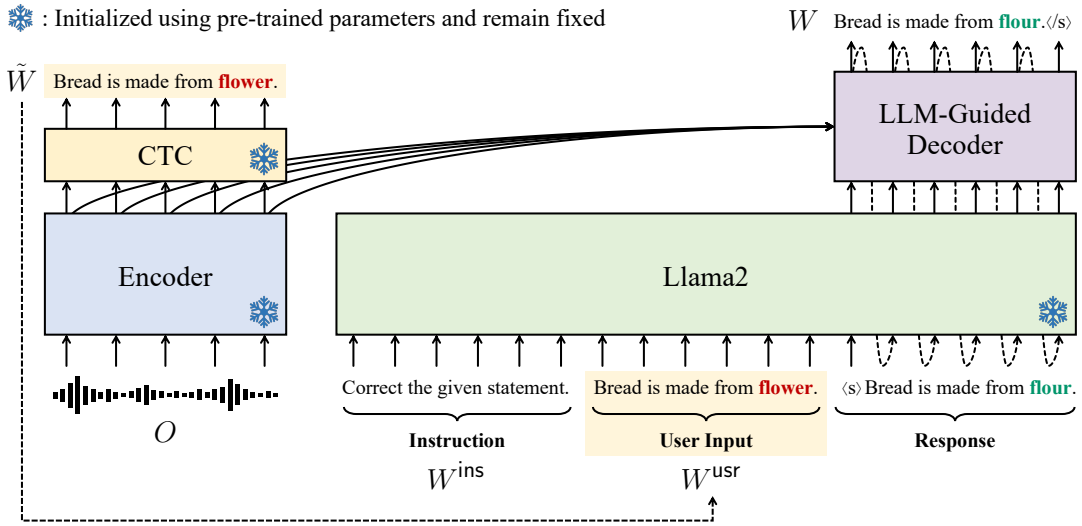

Figure 1: Overview of proposed end-to-end ASR model guided by instruction-tuned LLM, i.e., Llama2(-Chat). We construct a joint CTC/attention-based model, employing Llama2 as a front-end feature extractor for the decoder network. During inference, given a hypothesis $\tilde{W}$ generated via CTC decoding, Llama2 is tasked to perform grammatical error correction through precise prompting. The decoder network then produces an output sequence W, using text embeddings derived from Llama2 and aligning them with speech information via cross-attention.

sents autoregressive decoder layers, followed by a linear layer and the softmax function, which map to the output vocabulary, $\mathcal{V} \cup \{\texttt{</s>}\}$. Here, the decoder is equipped with the cross-attention mechanism for aligning each token in $W_{<n}$ to the encoder output H. The AED model is optimized by minimizing the negative log-likelihood of Eq. (4), $\mathcal{L}^{\text{aed}} \triangleq -\log p^{\text{aed}}(W|O)$.

CTC (Graves and Jaitly, 2014) formulates end-to-end ASR by evaluating all possible alignments between O and W. To align the sequences at the frame level, W is augmented by allowing repeated occurrences of the same token and inserting a blank symbol <b>. Let $A = (a_t \in \mathcal{V} \cup \{\texttt{<b>}\} | t = 1, \cdots, T')$ be an alignment sequence, and CTC models the posterior distribution of $p(W|O)$ as

$$p^{\text{ctc}}(W|O) \triangleq \sum_{A \in \mathcal{B}^{-1}(W)} \prod_{t=1}^{T'} p(a_t|O), \quad (7)$$

where $\mathcal{B} : A \mapsto W$ is a collapsing function that removes repeated tokens and blank symbols in A, and $\mathcal{B}^{-1}(W)$ represents a set of all possible alignments compatible with W. Using the encoder output H from Eq. (5), the token emission probability in Eq. (7) is computed as

$$p(a_t|O) = \text{CTC}(\mathbf{h}_t), \quad (8)$$

where $\text{CTC}(\cdot)$ represents a linear layer that maps to the output vocabulary of CTC, $\mathcal{V} \cup \{\varnothing\}$. The CTC model is optimized by minimizing the negative log-likelihood of Eq. (7), $\mathcal{L}^{\text{ctc}} \triangleq -\log p^{\text{ctc}}(W|O)$.

AED and CTC can be effectively combined to enhance robustness during training and inference processes of end-to-end ASR (Kim et al., 2017; Watanabe et al., 2017). The objective function of the joint model is defined as a linear interpolation of $\mathcal{L}^{\text{ctc}}$ and $\mathcal{L}^{\text{aed}}$ as

$$\mathcal{L}^{\text{ctc-aed}} = \lambda \mathcal{L}^{\text{ctc}} + (1-\lambda)\mathcal{L}^{\text{aed}}, \quad (9)$$

where λ ($0 \leq \lambda \leq 1$) is a tunable weight. Joint decoding is performed using a one-pass beam search, with CTC serving as a secondary score and the autoregressive decoder in AED primarily handles hypothesis expansion and end detection. The score of a hypothesis $\hat{W}$ is calculated using Eqs. (4) and (7) as $\xi \log p^{\text{ctc}}(\hat{W}|O) + (1-\xi) \log p^{\text{aed}}(\hat{W}|O)$, where ξ ($0 \leq \xi \leq 1$) is a tunable weight to define the importance of each score. See Hori et al. (2017) for a detailed decoding algorithm.

3 End-to-End Speech Recognition Guided by Instruction-Tuned LLM

Overview Figure 1 illustrates the proposed end-to-end ASR framework, which is specifically designed to leverage the text generation capabilities of an LLM, while ensuring proper alignment with speech information. This is achieved through the integration of an **LLM-guided decoder** into the joint CTC/attention framework (described in Sec. 2.2), where the LLM serves as a front-end feature extractor for the decoder, and the cross-attention mechanism within the decoder aligns the LLM-derived

features with the speech information embedded by the encoder. The LLM-guided decoder allows for accurate text generation by effectively incorporating the LLM knowledge, where the alignment with the speech information prevents *hallucinations* (e.g., the generation of unspoken words), which can be associated with standalone LLM outputs. To optimally derive linguistic features that facilitate the text generation process in the decoder, we capitalize on the LLM's potential as a zero-shot grammatical error correction model (Wu et al., 2023a; Fang et al., 2023; Yang et al., 2023), designing an effective prompting strategy that uses a hypothesized output sequence obtained by CTC.

The following subsections delve into the details of the proposed model, presenting a precise formulation that substantiates the effectiveness of our model design, which is followed by descriptions of training and inference strategies.

3.1 Formulation

The proposed model formulates end-to-end ASR by factorizing the posterior distribution $p(W|O)$ as

$$p(W|O) = \sum_{\tilde{W} \in \mathcal{H}(W)} p(W|\tilde{W}, O) p(\tilde{W}|O), \quad (10)$$

where $\tilde{W} \in \mathcal{V}^M$ is an M-length hypothesized output sequence, and $\mathcal{H}(W)$ represents a set of all possible output sequences compatible with W. In other words, $\mathcal{H}(W)$ comprises sequences that are prone to be misrecognized from input speech O, with $\tilde{W}$ derived from $p(\tilde{W}|O)$. In Eq. (10), we further factorize $p(W|\tilde{W}, O)$ by applying a probabilistic chain rule as

$$p(W|\tilde{W}, O) = \prod_{n=1}^{N+1} p(w_n|\tilde{W}, W_{<n}, O). \quad (11)$$

Eq. (11) follows the same formulation as AED in Eq. (4), but it additionally conditions the token emission probability on the hypothesized output $\tilde{W}$. Intuitively, Eq. (11) is interpreted as a model that estimates each current token based on previously predicted tokens $W_{<n}$, while also *correcting errors present in the hypothesized output sequence* $\tilde{W}$.

The token emission probability in Eq. (11) is modeled similarly to the AED architecture, with a modification to the decoder (i.e., Eq. (6)) as

$$\mathbf{e}_n = \text{Llama2}(W^{\text{ins}}, \tilde{W}, W_{<n}), \quad (12)$$

$$p(w_n|\tilde{W}, W_{<n}, O) = \text{LLMGuidedDecoder}(\mathbf{e}_1, \cdots, \mathbf{e}_n, H), \quad (13)$$

where H is the encoder output, as derived from Eq. (5). In Eq. (12), the Llama2 output $\mathbf{e}_n$ is obtained as in Eq. (1), where a hypothesized output sequence is used as the user input, i.e., $W^{\text{usr}} = \tilde{W}$, accompanied by an instruction W^{ins} that directs the LLM toward the grammatical error correction task (see Appendix B.4 for the actual prompt). Such a prompting strategy is expected to facilitate the modeling of Eq. (11). Eq. (13) represents the LLM-guided decoder, the key component of the proposed model, which is identical to the standard decoder in Eq. (6) but takes the Llama2 outputs $(\mathbf{e}_1, \cdots, \mathbf{e}_n)$ as input to align them with the encoder output H.

3.2 Inference

The most probable output sequence $\hat{W}$ is estimated by solving Eq. (10) as

$$\hat{W} = \operatorname*{argmax}_W \sum_{\tilde{W} \in \mathcal{H}(W)} p(W|\tilde{W}, O) p(\tilde{W}|O), \quad (14)$$

$$\approx \operatorname*{argmax}_W p(W|\tilde{W}', O), \quad (15)$$

$$\text{where } \tilde{W}' = \operatorname*{argmax}_{\tilde{W}} p(\tilde{W}|O). \quad (16)$$

To handle the intractable summation over $\tilde{W}$ in Eq. (14), we apply the Viterbi approximation with respect to $p(\tilde{W}|O)$, which results in Eqs. (15) and (16). The search process in Eq. (16) is implemented by the best path decoding of CTC (Graves et al., 2006) using Eq. (8), which first finds the most probable alignment $\hat{A}$ by concatenating the most active tokens at each time frame, $\hat{a}_t = \operatorname{argmax}_{a_t} p(a_t|O)$, and then obtains $\tilde{W}'$ by applying the collapsing function to $\hat{A}$ as $\tilde{W}' = \mathcal{B}(\hat{A})$. Eq. (15) is solved by performing the joint CTC/attention decoding (Hori et al., 2017; Watanabe et al., 2017) using scores derived from Eqs. (7) and (11), which results in the final output sequence $\hat{W}$. See Appendix A for the pseudocode of the proposed inference algorithm.

3.3 Training

The training process for the proposed model unfolds in two steps:

1. Train a joint CTC/attention model using $\mathcal{L}^{\text{ctc-aed}}$, as described in Eq. (9); and

2. Using the Encoder$(\cdot)$ and CTC$(\cdot)$ trained in Step 1 and the pre-trained Llama2$(\cdot)$, train LLMGuidedDecoder$(\cdot)$ in Eq. (13). Here, the LLM-guided decoder is only trained while the parameters of the other models remain fixed.

We freeze all the pre-trained networks in Step 2 not only to enhance training efficiency but also to enable the LLM-guided decoder to focus exclusively on aligning LLM-derived features with the speech information from the encoder.

In Step 2, the objective function of the proposed model is defined by the negative log-likelihood of Eq. (10) expanded with Eq. (11),

$$-\log \sum_{\tilde{W}\in\mathcal{H}(W)} \prod_{n=1}^{N+1} p(w_n|\tilde{W}, W_{<n}, O)p(\tilde{W}|O) \quad (17)$$

$$\approx -\log \mathbb{E}_{\tilde{W}\sim p(\tilde{W}|O)} \left[\prod_{n=1}^{N+1} p(w_n|\tilde{W}, W_{<n}, O)\right] \quad (18)$$

$$\leq \underbrace{-\mathbb{E}_{\tilde{W}\sim p(\tilde{W}|O)} \left[\log \prod_{n=1}^{N+1} p(w_n|\tilde{W}, W_{<n}, O)\right]}_{\triangleq \mathcal{L}^{\text{llm-dec}}}. \quad (19)$$

The intractable marginalization over $\tilde{W}$ in Eq. (17) is approximated by computing the expectation with respect to the sampling distribution based on the probability distribution of $p(\tilde{W}|O)$, which results in Eq. (18). By applying Jensen's inequality, the upper bound of Eq. (18) is derived to define the model's objective function $\mathcal{L}^{\text{llm-dec}}$ in Eq. (19). Practically, $\mathcal{L}^{\text{llm-dec}}$ is computed in a manner similar to the AED loss $\mathcal{L}^{\text{aed}}$, calculating the cross-entropy losses at each token prediction. The sampling process of $\tilde{W}$ in Eq. (19) is implemented by running the encoder in "training mode" (with dropout enabled) and performing the best path decoding of CTC, which is a similar strategy utilized in uncertainty estimation (Gal and Ghahramani, 2016; Vyas et al., 2019).

4 Additional Related Work

LLMs with Speech Input Prior studies have explored the incorporation of speech information in LLMs (Wang et al., 2023; Deshmukh et al., 2023), primarily focusing on enabling LLMs to accept speech input. In Wang et al. (2023), a single, shared decoder-only LLM has shown promising potential for being trained on both speech and text tasks. Similarly, other studies have successfully adapted pre-trained LLMs for speech tasks by designing an effective module for converting speech into the input space of LLMs. This process encodes speech into discrete tokens, typically derived from pre-trained acoustic models (Zhang et al., 2023; Rubenstein et al., 2023). Alternatively, pre-trained ASR models are employed to compress speech into a more manageable length (Chen et al., 2023a; Nachmani et al., 2024; Wu et al., 2023b; Yu et al., 2024; Fathullah et al., 2024). Our approach to leveraging LLMs for speech tasks is related to these studies, but differs conceptually in that we do not seek to directly adapt LLMs to speech. Instead, the proposed model is designed to extract linguistic information from LLMs that contributes to improving ASR performance, without requiring fine-tuning or modifications to pre-trained LLMs.

LM Integration in End-to-End ASR It has been a widely adopted practice to use separate LMs to improve the performance of end-to-end ASR systems. The traditional approach includes rescoring (Mikolov et al., 2010; Chan et al., 2016), which applies an LM score to the top N-best hypotheses generated by an ASR model. More advanced methods include incorporating LMs into beam search or directly into the model architectures, through techniques like shallow fusion (Hannun et al., 2014; Gulcehre et al., 2015), deep fusion (Gulcehre et al., 2015), and cold fusion (Sriram et al., 2018; Shan et al., 2019). These conventional approaches mainly focus on enhancing ASR models with LM information at the output probability or feature level. Our approach, in contrast, integrates an LLM into the input of the decoder to guide the text generation process. Nonetheless, we show that the proposed model remains compatible with conventional LM integration techniques.

Two-Pass End-to-End ASR In ASR, it is common to employ a second-pass model to refine outputs produced by a first-pass model (Sundermeyer et al., 2015; Chan et al., 2016; Kannan et al., 2018; Salazar et al., 2020). Recent advances in deep learning have enabled an ASR model to train both the first-pass and second-pass models in an end-to-end fashion, introducing an additional structure that refines a first-pass sequence (Xia et al., 2017; Wang et al., 2022; Higuchi et al., 2023b). The two-pass end-to-end ASR framework (Sainath et al., 2019) involves training a transducer-based model in conjunction with an attention decoder, which is specifically optimized to rescore hypotheses generated during transducer decoding. Additionally, acoustic embeddings from the encoder can help facilitate the training of the rescoring decoder (Hu et al., 2020). The proposed formulation in Eq. (10) shares similarities with these two-pass approaches. However, it differs in that the decoder does not specifically deliberate on hypotheses to generate an output sequence. Instead, it leverages a hypothesis to derive linguistic information from the LLM,

while learning to align this knowledge with speech information from the encoder output.

5 Experimental Setup

We used the ESPnet toolkit (Watanabe et al., 2018) for conducting our experiments, and all the codes and recipes will be publicly available for reproducibility.

Data We examined the effectiveness of our proposed approach on English ASR tasks, using various corpora spanning different amounts of data and domains, including LibriSpeech (LS) (Panayotov et al., 2015), TED-LIUM2 (TED2) (Rousseau et al., 2014), and CoVoST2 (CV2) (Wang et al., 2021). In addition to the full 960-hour training set in LS (LS-960), we used the 100-hour set (LS-100) to explore a lower-resource scenario and conduct further investigations and analyses. We specifically used CV2 for training models with punctuation and casing preserved, as this can be crucial for the LLM to accurately capture linguistic information. Full dataset descriptions and pre-processing details are in Appendix B.1.

Modeling We developed our baseline models within the joint CTC/attention framework (as detailed in Sec. 2.2), which used the Conformer-based architecture (Gulati et al., 2020; Guo et al., 2021). This baseline model also corresponds to the joint CTC/attention model trained in Step 1 in Sec. 3.3. The proposed model was constructed by substituting the decoder of the baseline model with our LLM-guided decoder (as described in Step 2 in Sec. 3.3). For the LLM, we used Llama2-Chat with 7B parameters (Touvron et al., 2023b), which was accessed through the HuggingFace library (Wolf et al., 2020). Full descriptions of model sizes, network architectures, and hyperparameters are in Appendix B.2.

Training and Decoding We primarily followed training/decoding configurations provided by the ESPnet recipes for each dataset. We set λ (in Eq. (9)) to 0.3 during baseline model training. For both the baseline and proposed models, we consistently set the score weight ξ to 0.3 during the joint CTC/attention decoding, unless specified otherwise. The beam size B was set to either 1 or 20. Notably, with $B = 1$, the influence of the LLM is more directly reflected in the proposed models. Descriptions of detailed configurations are in Appendix B.3.

Prompting We heuristically designed a prompt to guide Llama2 in performing grammatical error correction, setting W^{ins} (in Eq. (12)) to "*You will be provided with a statement in quotes. Correct the wrong words and provide your revised version.*". The specific format used and the process of determining the prompt are described in Appendix B.4.

Evaluation We measured ASR performance using the word error rate (WER). We also used the bootstrap method (Davison and Hinkley, 1997) to measure the statistical significance of the performance gains, which computes 95% confidence intervals for the difference in per-sample WER on a test set between the baseline and proposed models. Detailed configurations for computing the confidence intervals are provided in Appendix C.

6 Results and Analyses

6.1 Main Results

Table 1 presents a comparison of the baseline and proposed models across all the tasks, evaluating their performance based on the WER.[2] The `A0` results represent the CTC decoding performance, with ξ set to 1.0 to rely solely on the CTC score. Setting ξ to 0.3 for the joint decoding resulted in overall improvements across the tasks (`A0` vs. `A1`). The proposed model, featuring the LLM-guided decoder, achieved the best overall performance among the results with $B = 1$ (`A0`, `A1` vs. `A2`). This indicates the successful incorporation of the LLM capabilities into the text generation process, while only requiring the retraining of minimal parameters (e.g., 18.8M). In the proposed inference algorithm, as described in Sec. 3.2, the CTC decoding results (`A0`) served as inputs to Llama2 (i.e., $\tilde{W}$ in Eq. (12)). Thus, the gains from `A0` suggest that the proposed model effectively used the LLM to recover errors in the hypothesized outputs, as modeled in Eq. (11). We analyze the significance of the LLM in Sec. 6.2. In CV2, the proposed model demonstrated a notably higher level of improvement compared to those observed in the other tasks, particularly considering the gains from the CTC decoding results in `A0`. We attribute this to the use of unnormalized written-style text, which enabled the LLM to extract precise linguistic information.

[2]Note that our results from the joint CTC/attention models do not fully replicate the recent results reported in ESPnet (Peng et al., 2023). We attribute this discrepancy to the use of the Llama2 vocabulary in Eq. (6), which may not be optimal for end-to-end ASR training (Higuchi et al., 2023a).

ID	Model	B	ξ	LibriSpeech-100h Dev WER clean	LibriSpeech-100h Dev WER other	LibriSpeech-100h Test WER clean	LibriSpeech-100h Test WER other	LibriSpeech-960h Dev WER clean	LibriSpeech-960h Dev WER other	LibriSpeech-960h Test WER clean	LibriSpeech-960h Test WER other	TED-LIUM2 Dev WER	TED-LIUM2 Test WER	CoVoST2 Dev WER	CoVoST2 Test WER
A0	Joint CTC/Attention	1	1.0	9.3	21.6	9.7	22.2	3.0	7.3	3.2	7.3	10.0	9.3	23.7	26.3
A1	Joint CTC/Attention	1	0.3	9.9	20.6	10.7	21.1	3.2	**6.5**	3.4	**6.7**	11.6	8.8	18.9	21.8
A2	+ LLM-Guided Decoder	1	0.3	**6.7**	**17.5**	**7.3**	**17.9**	**2.6**	6.9	**2.8**	7.0	**9.5**	**7.8**	**15.6**	**18.1**
A3	Joint CTC/Attention	20	0.3	7.2	17.5	7.5	18.0	**2.3**	**5.7**	**2.6**	**5.7**	9.4	7.8	16.2	18.4
A4	+ LLM-Guided Decoder	20	0.3	**6.2**	**16.5**	**6.7**	**16.9**	2.6	6.8	2.8	7.0	**7.6**	**7.2**	**15.0**	**16.9**

Table 1: WERs [%] (↓) of our models with LLM-guided decoder compared to joint CTC/attention baselines. B and ξ denote the beam size and score weight, respectively, the parameters used during the joint CTC/attention decoding.

ID	Model	Dev WER $B=1$ clean	Dev WER $B=1$ other	Dev WER $B=20$ clean	Dev WER $B=20$ other
--	Joint CTC/Attn. (A1, A3)	9.9	20.6	7.2	17.5
--	+ LLM-Guided Dec. (A2, A4)	**6.7**	**17.5**	**6.2**	**16.5**
B1	w/o LLM	9.3	20.7	7.1	17.8
B2	w/o Prompt	9.3	19.7	6.5	16.7
B3	w/ Mismatched Task Inst.	6.8	17.8	6.5	16.8

Table 2: Ablation studies on LS-100, validating effectiveness of LLM and prompt in our proposed model.

When we set B to 20, the proposed model consistently outperformed the baseline across all tasks except for LS-960 (A3 vs. A4). Beam search decoding did not yield any performance gains in LS-960, potentially because Llama2 is more confident in its predictions for common and generic words within its vocabulary. Notably, we observed that our model generally struggles to recognize infrequent words (e.g., personal names), which are considered long-tail words rarely encountered during the pre-training process of Llama2. We further discuss these observations in Sec. 6.3.

For most of the tasks, we confirmed the significance of our performance gains by calculating the confidence intervals. The proposed model exhibited marginal improvements on LS-960, suggesting that the dataset already contained adequate text data for the model to accurately model the text generation process, thus reducing the reliance on the LLM. See Appendix C for detailed results.

6.2 Ablation Study

We conducted several ablation studies for the proposed model to assess the effectiveness of both the use of the LLM and the prompt. Table 2 presents the results of these ablation studies, evaluated by the WER on the LS-100 task.

Importance of LLM We ablated the LLM from the proposed model (B1), where, in Step 2 of the training process in Sec. 3.3, we trained the decoder from scratch but without using Llama2 as its front end. Compared to the baseline results (A1, A3), this modified training resulted in improvements in the "clean" set. However, there was a slight decline in performance on the "other" set, indicating a decrease in generalizability. With the integration of the LLM, the proposed model achieved significantly better results with superior generalization ability (A2, A4 vs. B1).

Influence of Prompt First, we trained the LLM-guided decoder by removing the prompt from the Llama2 input (B2), i.e., setting $W^{\mathsf{ins}} = \varnothing$ and $\tilde{W} = \varnothing$ in Eq. (12). While this modification resulted in modest gains compared to the baseline model (A1, A3 vs. B2), it adversely affected the performance of the proposed model (A2, A4 vs. B2). Next, we trained the LLM-guided decoder with a task instruction that diverged from grammatical error correction (B3), adapting W^{ins} to specify a translation task. See Appendix B.4 for the actual prompt used. The modified prompt yielded improvements over the baseline model (A1, A3 vs. B3), but the proposed model with the proper prompt demonstrated superior performance (A2, A4 vs. B2). The findings from B2 and B3 suggest that designing an appropriate prompt is crucial in our model to effectively leverage the LLM's capabilities, thereby optimizing its ability to extract linguistic information aligned with the target task.

6.3 Error Analysis with Decoding Example

Table 3 presents actual decoding results obtained from the baseline and proposed models on the LS test-other set. Comparing the results from the first utterance, it is observed that the CTC decoding (A0) generated non-words that can sound similar to the reference words. By performing the joint decod-

Joint CTC/Attn. (A0)	... by the passion of sympathy it had crd her into as many creases as an all glazed distended glo
Joint CTC/Attn. (A1)	... by the passion of sympathy it had crumbled her into as many creases as an old glowed distended bluff
+ LLM-Guided Dec. (A2)	... by the passion of sympathy it had crumpled her into as many creases as an old glazed distended glove
Reference	... by the passion of sympathy it had crumpled her into as many creases as an old glazed distended glove
Joint CTC/Attn. (A1)	ransom was pleased with the vision of that trinity it must be repeated that he was very provincible
+ LLM-Guided Dec. (A2)	frances was pleased with the vision of that remedy it must be repeated that he was very provincial
Reference	ransom was pleased with the vision of that remedy it must be repeated that he was very provincial

Table 3: Decoded samples (with speaker ID of *6128-63244*) in LS test-other set. Red indicates incorrect words.

	Dev WER		Test WER	
Integration Method	clean	other	clean	other
Joint CTC/Atten. (A3)	7.2	17.5	7.5	18.0
+ Shallow Fusion	6.4	17.1	7.0	17.5
+ Rescoring	**6.1**	**15.6**	**6.4**	**16.1**
+ Zero-Shot GEC	13.8	22.8	13.4	23.3
+ LLM-Guided Dec. (A4)	6.2	16.5	6.7	16.9

Table 4: WERs [%] (↓) for joint CTC/attention model using various LLM integration methods on LS-100.

	Test WER			
Model	LS-100	LS-960	TED2	CV2
Joint CTC/Atten. (A3)	7.5 / 18.0	2.6 / 5.7	7.8	18.4
+ Rescoring	6.4 / 16.1	**2.3** / **5.1**	7.3	15.8
+ LLM-Guided Dec. (A4)	6.7 / 16.9	2.8 / 7.0	7.2	16.9
+ Rescoring	**5.8** / **15.1**	2.4 / 6.0	**6.8**	**14.5**

Table 5: WERs [%] (↓) for proposed model across all ASR tasks, enhanced with LLM-based rescoring.

ing (A1), the baseline model effectively considered dependencies among subword outputs, resulting in the formation of actual words; however, some words remained contextually inappropriate. In contrast, the proposed model (A2), guided by the LLM, succeeded in generating accurate and semantically appropriate words. Analyzing the results from the second utterance, the proposed model predicted words more accurately than the baseline (A1 vs. A2), consistent with the observations from the first example. However, it is noteworthy that the model faced difficulties in properly recognizing the personal name "ransom." This tendency was similarly observed in other tasks as well.

6.4 Comparison with Conventional Language Model Integration Methods

Conventional approaches to integrating a separate LM during the inference of end-to-end ASR have also shown promising results when using LLMs (Udagawa et al., 2022; Hu et al., 2023). Table 4 presents WERs for the LS-100 task, reporting the performance of the baseline joint CTC/attention model when decoded using an LLM (i.e., Llama2). Specifically, we compare results obtained by performing shallow fusion, rescoring, and zero-shot grammatical error correction (GEC). Detailed descriptions for each method are in Appendix D. Looking at the results, both shallow fusion and rescoring resulted in notable performance improvements, with rescoring yielding larger gains than the proposed approach. Zero-shot GEC appeared to be challenging, as the LLM tended to produce hallucinations or overcorrections. These issues led to generating words not present in the speech input, indicating a need for a dedicated mechanism to align the LLM outputs with the speech information.

As the above-mentioned methods are specifically designed for use during inference, they can complement the proposed model, which integrates the LLM directly into the decoder network. To validate this, we focus on combining our model with the most promising rescoring method. Table 5 shows the results on all the tasks, demonstrating that the LLM-based rescoring further enhanced the performance of our proposed model.

7 Conclusion

We proposed to use an instruction-tuned LLM for guiding the text generation process in end-to-end ASR. Employing the LLM as a front-end module in the decoder, our model leveraged the LLM's powerful text generation capabilities, while cross-attention facilitated the alignment of LLM-derived features with the speech inputs. Additionally, we designed a prompting strategy to extract linguistic features beneficial for the decoder, utilizing a hypothesized output sequence generated via CTC decoding. Experimental results confirmed the effectiveness of the LLM, with the proposed model outperforming the joint CTC/attention baseline across major ASR tasks. Further investigations revealed the limitation of our model in recognizing domain-specific words and highlighted that our approach complements traditional LM integration methods.

Limitations

High Computational Cost A key limitation of the proposed model is its high computational requirements. This is due to the intensive forward computations in the LLM, which has large model size and carries out autoregressive sequence generation with a complexity of $\mathcal{O}(N^{\mathsf{ins}} + N^{\mathsf{usr}} + N)$. Although our approach reduces computational costs by eliminating the need for backward propagation through the LLM, it still requires a GPU with a large memory size (e.g., a single A100 with 40GB) for both the training and decoding phases. See Appendix E for a comparison of inference speeds. Future research should explore the application of compressed or lightweight LLMs, which are currently active areas of study in the field of NLP (Hsieh et al., 2023; Zhu et al., 2023; Ma et al., 2024).

Suboptimal Prompt We also recognize that the prompt used in our model is not ideal, as it was designed heuristically based on our empirical observations (refer to Appendix B.4) within the constraints of limited computing budgets. To further improve the proposed model, the prompt can be extended to enable few-shot in-context learning (Brown et al., 2020) or zero-shot reasoning (Kojima et al., 2022) for grammatical error correction, which has proven effective in previous studies (Ma et al., 2023b; Yang et al., 2023). This approach could bias the LLM more closely to the target domain, potentially addressing the issue of recognizing domain-specific words (as discussed in Secs. 6.1 and 6.3). Alternatively, if computational resources allow for more intensive backward computations, the prompt can be optimized jointly with the decoder by using a soft prompt (Lester et al., 2021; Song et al., 2023).

Non-Streaming Finally, we note that the proposed model is not suited for online streaming scenarios, as it requires a full-sentence hypothesis for the input to the LLM (i.e., $\tilde{W}$ in Eq. (12)). It is not particularly problematic for utterance-level ASR tasks. However, the proposed models face challenges in real-time applications like spoken dialogue systems that require immediate interaction. A viable solution to overcoming this limitation is to implement a two-pass streaming system (Sainath et al., 2019). Here, the encoder with CTC is built as a streaming model to produce real-time outputs, while the LLM-guided decoder operates after the utterance ends to refine the initial results.

References

Rosana Ardila, Megan Branson, Kelly Davis, Michael Kohler, Josh Meyer, Michael Henretty, Reuben Morais, Lindsay Saunders, Francis Tyers, and Gregor Weber. 2020. Common voice: A massively-multilingual speech corpus. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4218–4222.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *Proceedings of Advances in neural information processing systems 33*, pages 1877–1901.

William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. 2016. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Proceedings of the 2016 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4960–4964.

Feilong Chen, Minglun Han, Haozhi Zhao, Qingyang Zhang, Jing Shi, Shuang Xu, and Bo Xu. 2023a. X-LLM: Bootstrapping advanced large language models by treating multi-modalities as foreign languages. *arXiv preprint arXiv:2305.04160*.

Tongzhou Chen, Cyril Allauzen, Yinghui Huang, Daniel Park, David Rybach, W Ronny Huang, Rodrigo Cabrera, Kartik Audhkhasi, Bhuvana Ramabhadran, Pedro J Moreno, et al. 2023b. Large-scale language model rescoring on long-form data. In *Proceedings of the 2023 IEEE International Conference on Acoustics, Speech and Signal Processing*.

Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *Proceedings of Advances in Neural Information Processing Systems 28*, pages 577–585.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. PaLM: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):11324–11436.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

Anthony Christopher Davison and David Victor Hinkley. 1997. *Bootstrap methods and their application*. 1. Cambridge University Press.

Keqi Deng, Zehui Yang, Shinji Watanabe, Yosuke Higuchi, Gaofeng Cheng, and Pengyuan Zhang. 2022. Improving non-autoregressive end-to-end speech recognition with pre-trained acoustic and language models. In *Proceedings of the 2022 IEEE*

International Conference on Acoustics, Speech and Signal Processing, pages 8522–8526.

Soham Deshmukh, Benjamin Elizalde, Rita Singh, and Huaming Wang. 2023. Pengi: An audio language model for audio tasks. In *Proceedings of Advances in Neural Information Processing Systems 36*, pages 18090–18108.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional Transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186.

Tao Fang, Shu Yang, Kaixin Lan, Derek F Wong, Jinpeng Hu, Lidia S Chao, and Yue Zhang. 2023. Is ChatGPT a highly fluent grammatical error correction system? a comprehensive evaluation. *arXiv preprint arXiv:2304.01746*.

Yassir Fathullah, Chunyang Wu, Egor Lakomkin, Junteng Jia, Yuan Shangguan, Ke Li, Jinxi Guo, Wenhan Xiong, Jay Mahadeokar, Ozlem Kalinli, et al. 2024. Prompting large language models with speech recognition abilities. In *Proceedings of the 2024 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 13351–13355.

Luciana Ferrer and Pablo Riera. Confidence intervals for evaluation in machine learning [computer software]. `https://github.com/luferrer/ConfidenceIntervals`. [Online; Accessed on June-1-2024].

Hayato Futami, Hirofumi Inaguma, Sei Ueno, Masato Mimura, Shinsuke Sakai, and Tatsuya Kawahara. 2020. Distilling the knowledge of BERT for sequence-to-sequence ASR. In *Proceedings of Interspeech 2020*, pages 3635–3639.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 1050–1059.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2021. A framework for few-shot language model evaluation.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 369–376.

Alex Graves and Navdeep Jaitly. 2014. Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the 31st International Conference on International Conference on Machine Learning*, pages 1764–1772.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. 2020. Conformer: Convolution-augmented Transformer for speech recognition. In *Proceedings of Interspeech 2020*, pages 5036–5040.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*.

Pengcheng Guo, Florian Boyer, Xuankai Chang, Tomoki Hayashi, Yosuke Higuchi, Hirofumi Inaguma, Naoyuki Kamo, Chenda Li, Daniel Garcia-Romero, Jiatong Shi, et al. 2021. Recent developments on ESPnet toolkit boosted by Conformer. In *Proceedings of the 2021 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5874–5878.

Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. 2014. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*.

Yosuke Higuchi, Tetsuji Ogawa, Tetsunori Kobayashi, and Shinji Watanabe. 2023a. BECTRA: Transducer-based end-to-end ASR with BERT-enhanced encoder. In *Proceedings of the 2023 IEEE International Conference on Acoustics, Speech and Signal Processing*.

Yosuke Higuchi, Andrew Rosenberg, Yuan Wang, Murali Karthick Baskar, and Bhuvana Ramabhadran. 2023b. Mask-Conformer: Augmenting conformer with mask-predict decoder. In *Proceedings of the 2023 IEEE Automatic Speech Recognition and Understanding Workshop*.

Yosuke Higuchi, Brian Yan, Siddhant Arora, Tetsuji Ogawa, Tetsunori Kobayashi, and Shinji Watanabe. 2022. BERT meets CTC: New formulation of end-to-end speech recognition with pre-trained masked language model. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5486–5503.

Takaaki Hori, Shinji Watanabe, and John R Hershey. 2017. Joint CTC/attention decoding for end-to-end speech recognition. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 518–529.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8003–8017.

Ke Hu, Tara N Sainath, Bo Li, Nan Du, Yanping Huang, Andrew M Dai, Yu Zhang, Rodrigo Cabrera, Zhifeng Chen, and Trevor Strohman. 2023. Massively multilingual shallow fusion with large language models. In *Proceedings of the 2023 IEEE International Conference on Acoustics, Speech and Signal Processing*.

Ke Hu, Tara N Sainath, Ruoming Pang, and Rohit Prabhavalkar. 2020. Deliberation model based two-pass end-to-end speech recognition. In *Proceedings of the 2020 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7799–7803.

Anjuli Kannan, Yonghui Wu, Patrick Nguyen, Tara N Sainath, Zhijeng Chen, and Rohit Prabhavalkar. 2018. An analysis of incorporating an external language model into a sequence-to-sequence model. In *Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5824–5828.

Suyoun Kim, Takaaki Hori, and Shinji Watanabe. 2017. Joint CTC-attention based end-to-end speech recognition using multi-task learning. In *Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4835–4839.

Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. 2015. Audio augmentation for speech recognition. In *Proceedings of Interspeech 2015*, pages 3586–3589.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Proceedings of Advances in Neural Information Processing Systems 35*, volume 35, pages 22199–22213.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.

Rao Ma, Mark J. F. Gales, Kate M. Knill, and Mengjie Qian. 2023a. N-best T5: Robust ASR error correction using multiple input hypotheses and constrained decoding space. In *Proceedings of Interspeech 2023*, pages 3267–3271.

Rao Ma, Mengjie Qian, Potsawee Manakul, Mark Gales, and Kate Knill. 2023b. Can generative large language models perform ASR error correction? *arXiv preprint arXiv:2307.04172*.

Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. 2024. The era of 1-bit LLMs: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of Interspeech 2010*, pages 1045–1048.

Eliya Nachmani, Alon Levkovitch, Roy Hirsch, Julian Salazar, Chulayuth Asawaroengchai, Soroosh Mariooryad, Ehud Rivlin, RJ Skerry-Ryan, and Michelle Tadmor Ramanovich. 2024. Spoken question answering and speech continuation using spectrogram-powered LLM. In *Proceedings of the 12th International Conference on Learning Representations*.

OpenAI. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. In *Proceedings of Advances in neural information processing systems 35*, pages 27730–27744.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: An ASR corpus based on public domain audio books. In *Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5206–5210.

Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. 2019. SpecAugment: A simple data augmentation method for automatic speech recognition. In *Proceedings of Interspeech 2019*, pages 2613–2617.

Daniel S Park, Yu Zhang, Chung-Cheng Chiu, Youzheng Chen, Bo Li, William Chan, Quoc V Le, and Yonghui Wu. 2020. SpecAugment on large scale datasets. In *Proceedings of the 2020 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6879–6883.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Proceedings of Advances in Neural Information Processing Systems 32*.

Yifan Peng, Kwangyoun Kim, Felix Wu, Brian Yan, Siddhant Arora, William Chen, Jiyang Tang, Suwon Shon, Prashant Sridhar, and Shinji Watanabe. 2023. A comparative study on E-Branchformer vs Conformer in speech recognition, translation, and understanding tasks. In *Proceedings of Interspeech 2023*, volume 2023, pages 2208–2212.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Anthony Rousseau, Paul Deléglise, and Yannick Estève. 2014. Enhancing the TED-LIUM corpus with selected data for language modeling and more TED talks. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*, pages 3935–3939.

Paul K Rubenstein, Chulayuth Asawaroengchai, Duc Dung Nguyen, Ankur Bapna, Zalán Borsos, Félix de Chaumont Quitry, Peter Chen, Dalia El Badawy, Wei Han, Eugene Kharitonov, et al. 2023. AudioPaLM: A large language model that can speak and listen. *arXiv preprint arXiv:2306.12925*.

Tara N. Sainath, Ruoming Pang, David Rybach, Yanzhang He, Rohit Prabhavalkar, Wei Li, Mirkó Visontai, Qiao Liang, Trevor Strohman, Yonghui Wu, Ian McGraw, and Chung-Cheng Chiu. 2019. Two-pass end-to-end speech recognition. In *Proceedings of Interspeech 2019*, pages 2773–2777.

Julian Salazar, Davis Liang, Toan Q Nguyen, and Katrin Kirchhoff. 2020. Masked language model scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2699–2712.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. BLOOM: A 176B-parameter open-access multilingual language model. *hal-03850124*.

Changhao Shan, Chao Weng, Guangsen Wang, Dan Su, Min Luo, Dong Yu, and Lei Xie. 2019. Component fusion: Learning replaceable language model component for end-to-end speech recognition system. In *Proceedings of the 2019 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5361–5635.

Gan Song, Zelin Wu, Golan Pundak, Angad Chandorkar, Kandarp Joshi, Xavier Velez, Diamantino Caseiro, Ben Haynor, Weiran Wang, Nikhil Siddhartha, Pat Rondon, and Khe Chai Sim. 2023. Contextual spelling correction with large language models. In *Proceedings of the 2023 IEEE Automatic Speech Recognition and Understanding Workshop*.

Anuroop Sriram, Heewoo Jun, Sanjeev Satheesh, and Adam Coates. 2018. Cold fusion: Training seq2seq models together with language models. In *Proceedings of Interspeech 2018*, pages 387–391.

Martin Sundermeyer, Hermann Ney, and Ralf Schlüter. 2015. From feedforward to recurrent LSTM neural networks for language modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):517–529.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Takuma Udagawa, Masayuki Suzuki, Gakuto Kurata, Nobuyasu Itoh, and George Saon. 2022. Effect and analysis of large-scale language model rescoring on competitive ASR systems. In *Proceedings of Interspeech 2022*, pages 3919–3923.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of Advances in Neural Information Processing Systems 30*, pages 6000–6010.

Apoorv Vyas, Pranay Dighe, Sibo Tong, and Hervé Bourlard. 2019. Analyzing uncertainties in speech recognition using dropout. In *Proceedings of the 2019 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6730–6734.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Changhan Wang, Anne Wu, Jiatao Gu, and Juan Pino. 2021. CoVoST 2 and massively multilingual speech translation. In *Proceedings of Interspeech 2021*, pages 2247–2251.

Tianrui Wang, Long Zhou, Ziqiang Zhang, Yu Wu, Shujie Liu, Yashesh Gaur, Zhuo Chen, Jinyu Li, and Furu Wei. 2023. VioLA: Unified codec language models for speech recognition, synthesis, and translation. *arXiv preprint arXiv:2305.16107*.

Weiran Wang, Ke Hu, and Tara N Sainath. 2022. Deliberation of streaming RNN-transducer by non-autoregressive decoding. In *Proceedings of the 2022 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7452–7456.

Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, Adithya Renduchintala, and Tsubasa Ochiai. 2018. ESPnet: End-to-end speech processing toolkit. In *Proceedings of Interspeech 2019*, pages 2207–2211.

Shinji Watanabe, Takaaki Hori, Suyoun Kim, John R Hershey, and Tomoki Hayashi. 2017. Hybrid CTC/attention architecture for end-to-end speech recognition. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1240–1253.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022a. Finetuned language models are zero-shot learners. In *Proceedings of the 10th International Conference on Learning Representations*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022b. Emergent abilities of large language models. *Transactions on Machine Learning Research*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

Haoran Wu, Wenxuan Wang, Yuxuan Wan, Wenxiang Jiao, and Michael Lyu. 2023a. ChatGPT or Grammarly? evaluating ChatGPT on grammatical error correction benchmark. *arXiv preprint arXiv:2303.13648*.

Jian Wu, Yashesh Gaur, Zhuo Chen, Long Zhou, Yimeng Zhu, Tianrui Wang, Jinyu Li, Shujie Liu, Bo Ren, Linquan Liu, et al. 2023b. On decoder-only architecture for speech-to-text and large language model integration. In *Proceedings of the 2023 IEEE Automatic Speech Recognition and Understanding Workshop*.

Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2017. Deliberation networks: Sequence generation beyond one-pass decoding. In *Proceedings of Advances in Neural Information Processing Systems 30*.

Chao-Han Huck Yang, Yile Gu, Yi-Chieh Liu, Shalini Ghosh, Ivan Bulyko, and Andreas Stolcke. 2023. Generative speech recognition error correction with large language models and task-activating prompting. In *Proceedings of the 2023 IEEE Automatic Speech Recognition and Understanding Workshop*.

Cheng Yi, Shiyu Zhou, and Bo Xu. 2021. Efficiently fusing pretrained acoustic and linguistic encoders for low-resource speech recognition. *IEEE Signal Processing Letters*, 28:788–792.

Algorithm 1 Inference algorithm

Input:
O ▷ Input speech
ξ ▷ Score weight for joint decoding
B ▷ Beam size

1: **function** BESTPATHDECODING(H)
2: // Obtain the most probable alignment using Eq. (8)
3: $\hat{A} \leftarrow \left(\hat{a}_t = \text{argmax}_{a_t}\, p(a_t|O) | t = 1, \cdots, T\right)$
4: $\tilde{W}' \leftarrow \mathcal{B}(\hat{A})$
5: **return** $\tilde{W}'$
6: **function** JOINTCTCATTDECODING(H, $\tilde{W}'$, ξ, B)
7: // Initialize a hypothesis set
8: $\Omega \leftarrow \{\}$
9: // Define the score function, where the CTC score is
10: // computed using Eqs. (7) and (8) and the attention
11: // score is computed using Eqs. (11) to (13)
12: $\text{SCORE}(\cdot) \coloneqq \xi \log p^{\text{ctc}}(\cdot|O) + (1-\xi) \log p(\cdot|\tilde{W}', O)$
13: $\Omega \leftarrow \text{BEAMSEARCH}(B, \text{SCORE}(\cdot))$
14: $\hat{W} \leftarrow \text{argmax}(\Omega)$
15: **return** $\hat{W}$
16: $H \leftarrow \text{Encoder}(O)$
17: $\tilde{W}' \leftarrow \text{BESTPATHDECODING}(H)$
18: $\hat{W} \leftarrow \text{JOINTCTCATTDECODING}(H, \tilde{W}', \xi, B)$
19: **return** $\hat{W}$

Wenyi Yu, Changli Tang, Guangzhi Sun, Xianzhao Chen, Tian Tan, Wei Li, Lu Lu, Zejun Ma, and Chao Zhang. 2024. Connecting speech encoder and large language model for ASR. In *Proceedings of the 2024 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 12637–12641.

Dong Zhang, Shimin Li, Xin Zhang, Jun Zhan, Pengyu Wang, Yaqian Zhou, and Xipeng Qiu. 2023. SpeechGPT: Empowering large language models with intrinsic cross-modal conversational abilities. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15757–15773.

Guolin Zheng, Yubei Xiao, Ke Gong, Pan Zhou, Xiaodan Liang, and Liang Lin. 2021. Wav-BERT: Cooperative acoustic and linguistic representation learning for low-resource speech recognition. In *Proc. Findings of EMNLP*, pages 2765–2777.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2023. A survey on model compression for large language models. *arXiv preprint arXiv:2308.07633*.

A Pseudocode for Proposed Inference Algorithm

Algorithm 1 presents the pseudocode for the proposed inference algorithm, as described in Sec. 3.2. The search process of Eq. (16) is implemented by BESTPATHDECODING($\cdot$) (line 17) and Eq. (14) is implemented by JOINTCTCATTDECODING($\cdot$) (line 18). The joint decoding process is the same as that in the baseline CTC/attention model, except

Data Split		#hours	#utterances
Training	train-clean-100	100	28k
	train-clean-360	360	104k
	train-other-500	500	149k
	train-960	960	281k
Development	dev-clean	5.4	2703
	dev-other	5.1	2864
Evaluation	test-clean	5.4	2620
	test-other	5.3	2939

Table 6: Dataset description of LibriSpeech.

Data Split		#hours	#utterances
Training	Train	207	93k
Development	Dev	1.6	507
Evaluation	Test	2.6	1155

Table 7: Dataset description of TED-LIUM2

that the score computation (line 12) is based on the LLM-guided decoder (Eq. (4) vs. Eq. (11)).

B Reproducibility

This section provides additional details on the experimental settings for reproducibility.

B.1 Data

B.1.1 Dataset Descriptions

LibriSpeech (LS) LS (Panayotov et al., 2015) consists of utterances derived from read English audiobooks. Table 6 lists the statistics for data splits included in LS. Each data split is categorized as "clean" or "other" based on the audio quality. The official development (*dev-clean* and *dev-other*) and evaluation (*test-clean* and *test-other*) sets were used for tuning hyper-parameters and evaluating performance, respectively. Data preparation was done using the ESPnet2 recipes[3].

TED-LIUM2 (TED2) TED2 (Rousseau et al., 2014) contains utterances from English TED Talks. Table 7 lists the statistics for data splits included in TED2. The official development (*Dev*) and evaluation (*Test*) sets were used for tuning hyper-parameters and evaluating performance, respectively. Data preparation was done using the ESPnet2 recipe[4].

[3] https://github.com/espnet/espnet/tree/master/egs2/{librispeech,librispeech_100}/asr1

[4] https://github.com/espnet/espnet/tree/master/egs2/tedlium2/asr1

Data Split		#hours	#utterances
Training	Train	407	272k
Development	Dev	22	13k
Evaluation	Test	25	16k

Table 8: Dataset description of CoVoST2 (En→X).

CoVoST2 (CV2) CV2 (Wang et al., 2021) is a corpus designed for speech translation tasks, derived from the Common Voice project (Ardila et al., 2020). We used CV2 as an English ASR task by exclusively extracting source speech-text data from the "En→X" task. Table 8 lists the statistics for data splits of CV2 used in our experiments. The official development (*Dev*) and evaluation (*Test*) sets were used for tuning hyper-parameters and evaluating performance, respectively. Data preparation was done using the ESPnet2 recipe[5].

B.1.2 Pre-Processing

The transcriptions provided by the above corpora, except for CV2, are normalized by default, where punctuation was removed, and casing was standardized to lowercase. For CV2, we used unnormalized transcriptions during training, with punctuation and casing preserved. During the evaluation on CV2, we removed punctuation from both the reference and hypothesis before computing the WER. All the text tokenization was done using the vocabulary of Llama2, where the vocabulary size $|\mathcal{V}|$ was 32k.

B.2 Modeling Details

The encoder, Encoder$(\cdot)$ in Eq. (5), consisted of two convolutional neural network (CNN) layers followed by a stack of 12 Conformer encoder blocks (Gulati et al., 2020). Each CNN layer had 256 channels, a kernel size of 3×3, and a stride size of 2, which resulted in down-sampling the input length by a factor of 4 (i.e., $T' = T/4$). In each encoder block, the number of head D^{head}, the dimension of a self-attention layer D^{asr}, the dimension of a feed-forward network D^{ff}, and the kernel size were set to (4, 256, 1024, 31) for LS-100, TED2, and CV2; and (8, 512, 2048, 31) for LS-960. The decoder, Decoder$(\cdot)$ in Eq. (6), was a stack of 6 Transformer decoder blocks (Vaswani et al., 2017), where (D^{head}, D^{asr}, D^{ff}) were set to (4, 256, 2048) for LS-100, TED2, and CV2; and

[5] https://github.com/espnet/espnet/tree/master/egs2/covost2/asr1

Task	Network	#params
LS-100 / TED2 / CV2	Encoder + CTC	29.1M
	Decoder	25.9M
	LLMGuidedDecoder	18.8M
	Llama2	6.7B
LS-960	Encoder + CTC	99.6M
	Decoder	58.0M
	LLMGuidedDecoder	43.7M
	Llama2	6.7B

Table 9: Number of parameters in model components.

Hyperparameter	Value
Hidden dropout rate	0.1
Attention dropout rate	0.1
Activation dropout rate	0.1
LR scheduling	Noam (Vaswani et al., 2017)
Peak learning rate	best of $\{1.5, 2.0\} \times 10^{-3}$
Warmup steps	best of $\{15\text{k}, 40\text{k}\}$
Adam betas	$(0.9, 0.999)$
Adam epsilon	10^{-8}
Weight decay rate	10^{-6}

Table 10: Training hyperparameters.

$(8, 512, 2048)$ for LS-960. The LLM-guided decoder, LLMGuidedDecoder$(\cdot)$ in Eq. (13), shared the same architecture as Decoder$(\cdot)$, but the embedding layer was replaced by a linear layer that converts the Llama2 output $\mathbf{e}_n \in \mathbb{R}^{D^{\text{llm}}}$ to a D^{asr}-dimensional vector, without positional encoding. The LLM used in the LLM-guided decoder, Llama2$(\cdot)$ in Eq. (12), was Llama2(-Chat) (Touvron et al., 2023b), which was downloaded from the HuggingFace library[6]. The dimension of the self-attention layer in Llama2 D^{llm} was 4096.

Table 9 lists the parameter counts for each model component. The baseline model contained 55.0M total and trainable parameters for LS-100, TED2, and CV2, and 157.7M for LS-960. The proposed model contained 6.7B total parameters for LS-100, TED2, and CV2, and 6.8B for LS-960. The trainable parameters were 18.8M for LS-100, TED2, and CV2, and 43.7M for LS-960.

B.3 Training and Decoding Configurations

All the models were implemented and trained using ESPnet (Watanabe et al., 2018)[7] and PyTorch (Paszke et al., 2019)[8]. The baseline models were trained up to 50 epochs (i.e., Step 1 in Sec. 3.3), and subsequently, the proposed models with the LLM-guided decoder were trained up to 50 epochs for LS-100, and 25 epochs for the other tasks (i.e., Step 2 in Sec. 3.3). We augmented speech data using speed perturbation (Ko et al., 2015) with a factor of 3 and SpecAugment (Park et al., 2019, 2020). For the hyperparameters in SpecAugment, we set the number of frequency and time masks to 2 and 5, and the size of frequency and time masks to 27 and $0.05T$. See Table 10 for other training hyperparameters. After training, a final model was obtained for evaluation by averaging model parameters over ten checkpoints with the best validation accuracy.

The baseline models were trained on four V100 (16GB) GPUs for 1 to 3 days, depending on the task. Decoding was performed using a single V100 GPU. The proposed models were trained on a single A100 (40GB) GPU for 1 to 6 days, depending on the task. Decoding was performed using a single A100 GPU.

B.4 Prompt Details

We followed the prompting format of Llama2 described in Touvron et al. (2023b), which resulted in the following prompt.

```
<s>[INST] <<SYS>>
You will be provided with a statement
in quotes. Correct the wrong words and
provide your revised version.
<</SYS>>

"${ASR_HYPOTHESIS}" [/INST]
```

Here, `${ASR_HYPOTHESIS}` corresponds to the hypothesized output sequence $\tilde{W}$ $(= W^{\text{usr}})$ obtained via CTC decoding. As specified in the task instruction, we enclosed the hypothesis in double quotation marks. This has been found crucial for the model to accurately identify the target sequence, as certain sequences have been observed to cause misinterpretations. For example, in CV2, a user input sequence like "Do you know anything about it?" led the LLM to produce a generic response.

We heuristically adjusted the prompt by using Llama2 exclusively to perform zero-shot grammatical error correction on ASR hypotheses, as discussed in Sec. 6.4. Through qualitative observation, we selected a prompt that adhered accurately to the specified task and minimized hallucinations, such as unnecessary rephrasing and the insertion of unspoken words, from the initial hypothesis. Ad-

[6] https://huggingface.co/meta-llama/Llama-2-7b-chat-hf

[7] https://github.com/espnet/espnet

[8] https://github.com/pytorch/pytorch

Experiments	Confidence Intervals [ΔWER]			
	LS-100	LS-960	TED2	CV2
A1 vs. A2	$(2.21\%, 2.82\%)$	$(-0.32\%, 0.11\%)$	$(0.66\%, 1.51\%)$	$(3.44\%, 3.91\%)$
A3 vs. A4	$(0.54\%, 0.90\%)$	$(-1.99\%, -1.24\%)$	$(0.23\%, 0.84\%)$	$(0.94\%, 1.52\%)$

Table 11: Confidence intervals calculated for test WER differences between baseline and proposed models. The experiment IDs correspond to those defined in Table 1.

ditionally, we aimed to keep the prompt concise to reduce memory usage during training.

In Table 2, experiment B3 was conducted by using the following prompt.

```
<s>[INST] <<SYS>>
You will be provided with a statement in
quotes, and your task is to translate it
into Japanese.
<</SYS>>

“${ASR_HYPOTHESIS}” [/INST]
```

C Confidence Intervals

We calculated the confidence intervals based on the bootstrap method, using the tool provided by Ferrer and Riera. Here, 95% confidence intervals were calculated for the difference in per-sample WER (ΔWER) on a test set between the baseline and proposed models, setting the number of bootstrap samples at 1000. Table 11 lists the confidence intervals calculated on the results presented in Table 1. If a confidence interval excludes the value 0.0, we can reject the null hypothesis that the two models have the same performance.

D Language Model Integration Methods

This section describes the LM integration techniques used to obtain the results in Tables 4 and 5.

D.1 Shallow Fusion

Shallow fusion was implemented by incorporating an LM score into the scoring process (i.e., line 12 in Algorithm 1) during the joint CTC/attention decoding. The LM score was computed using the Llama2 probability in Eq. (2) as

$$\log p^{\mathsf{lm}}(W) \triangleq \log p(W|W^{\mathsf{ins}}, W^{\mathsf{usr}}), \qquad (20)$$

where we specified W^{ins} using the same prompt used in the proposed model (see Appendix B.4). Similarly, we used the CTC decoding results (A0 in Table 1) for W^{usr}. During beam search, the score of an output sequence W was computed as

$$\begin{aligned}\log p(W|O) = {} & \xi \log p^{\mathsf{ctc}}(W|O) \\ & + (1-\xi) \log p^{\mathsf{aed}}(W|O) \\ & + \gamma \log p^{\mathsf{lm}}(W), \end{aligned} \qquad (21)$$

where γ represents the weight for the LM score, and we tuned γ from $\{0.1, 0.3, 0.5, 0.7, 1.0\}$.

D.2 Rescoring

Rescoring was implemented by using the LM score to rerank the hypotheses, i.e., Ω in Algorithm 1, obtained from the joint decoding. Specifically, the top-K hypotheses were first extracted from Ω to form Ω', and then the most probable sequence $\hat{W}$ was selected from Ω' as

$$\begin{aligned}\hat{W} = \underset{W \in \Omega'}{\operatorname{argmax}} \Big(& \xi \log p^{\mathsf{ctc}}(W|O) \\ & + (1-\xi) \log p^{\mathsf{aed}}(W|O) \\ & + \omega \log p^{\mathsf{lm}}(W) \Big), \end{aligned} \qquad (22)$$

where ω denotes the weight for the LM score derived from Eq. (20). Unlike shallow fusion, we computed the LM score in Eq. (22) without the prompt, setting both W^{ins} and W^{usr} to $\varnothing$, as it did not affect the performance. We tuned K from $\{5, 10, 15\}$ and ω from $\{0.1, 0.3, 0.5, 0.7, 1.0\}$, where $K = 10$ and $\omega = 0.5$ consistently delivered reasonable results across all tasks.

D.3 Zero-Shot Grammatical Error Correction

Zero-shot grammatical error correction was conducted by evaluating the outputs of the standalone LLM, which were generated based on the Llama2 probability in Eq. (2). For W^{ins}, we specified the same task instruction used in the proposed model (see Appendix B.4). For W^{usr}, we used the results from the joint CTC/attention model (A3 in Table 1). During the generation process, we configured the maximum output sequence length to 512. We also performed the beam search decoding with a beam size tuned from $\{5, 10, 20\}$, where we did not em-

Model	B	RTF
Joint CTC/Attention (A1)	1	0.147
+ LLM-Guided Decoder (A2)	1	0.222
Joint CTC/Attention (A3)	20	0.194
+ LLM-Guided Decoder (A4)	20	1.193

Table 12: RTF (↓) of our model with LLM-guided decoder compared to joint CTC/attention baseline.

ploy any sampling strategies. To ensure Llama2 exclusively generates its corrected sequence, we appended a double quotation mark (") immediately after the prompt. This was based on observations that the model frequently encloses its corrected sequences within double quotation marks.

E Inference Speed Comparison

Table 12 compares the inference speeds of the baseline and proposed models, using the real-time factor (RTF). RTF was measured on the LS test-other set using a single A100 GPU with a batchsize of 1. These results represent the average values obtained from three separate runs.