SALSA: Speedy ASR-LLM Synchronous Aggregation

Ashish Mittal[*1,2], *Darshan Prabhu*[*2], *Sunita Sarawagi*[2], *Preethi Jyothi*[2]

[1]IBM Research, [2]IIT Bombay

arakeshk@in.ibm.com, {darshanp, sunita, pjyothi}@cse.iitb.ac.in

Abstract

Harnessing pre-trained LLMs to improve ASR systems, particularly for low-resource languages, is now an emerging area of research. Existing methods range from using LLMs for ASR error correction to tightly coupled systems that replace the ASR decoder with the LLM. These approaches either increase decoding time or require expensive training of the cross-attention layers. We propose SALSA, which couples the decoder layers of the ASR to the LLM decoder, while synchronously advancing both decoders. Such coupling is performed with a simple projection of the last decoder state, and is thus significantly more training efficient than earlier approaches. A challenge of our proposed coupling is handling the mismatch between the tokenizers of the LLM and ASR systems. We handle this mismatch using cascading tokenization with respect to the LLM and ASR vocabularies. We evaluate SALSA on 8 low-resource languages in the FLEURS benchmark, yielding substantial WER reductions of up to 38%.

Index Terms: speech recognition, large language models, low-resource languages

1. Introduction

Automatic speech recognition (ASR) systems, whether they are cascaded or end-to-end, have been shown to benefit from both a strong acoustic model and a strong language model. When the acoustic model is inadequate, particularly for low-resource languages with limited access to labeled speech, the language model can offer effective supplementary support [1, 2]. With the advent of pretrained large language models (LLMs) and their superior text modeling abilities, there is growing interest in leveraging LLMs to improve ASR performance [3, 4, 5]. This LLM-ASR integration could be particularly beneficial for low-resource languages on which the ASR model underperforms and for which the LLM model has sufficient base capabilities.

There are broadly three ways in which LLMs have been leveraged for ASR in recent work.

1. ASR error correction: The ASR system generates N-best lists and the LLM rescores them. This could be achieved by prompting the LLM with the N-best list and attending to the speech encoder [6, 7].
2. Speech in-context learning: Pretrained LLMs are additionally instruction-tuned with speech inputs enabling a tight coupling between a speech encoder and the LLM to support multiple speech tasks including ASR [3, 8, 9].
3. Deep LLM-fusion: The LLM replaces the decoder of an encoder-decoder ASR system via gated cross-attention [5].

All these prior approaches are computationally expensive with high training overhead in the deep LLM-fusion and speech in-context learning paradigms due to fine-tuning, and high decoding latency due to second-pass rescoring in ASR error correction. In this work, we propose a lightweight alternative called SALSA[1] that offers a deeper integration of LLMs with ASR beyond shallow fusion while incurring a low training overhead. To the best of our knowledge, we are also the first to show the utility of LLMs for ASR of a diverse set of low-resource languages; all prior work in this area has focused on English ASR.

In SALSA, we keep the pretrained ASR and LLM backbone architectures frozen and only train feedforward projection layers that couple the ASR decoder layers to the LLM decoder layers. SALSA requires that both ASR and LLM decoders move forward in tandem albeit having different tokenizations. The LLM autoregressively predicts the next token, and as soon as a valid ASR tokenizable text is formed, advances the ASR decoder with the predicted text. With each synchronized step, the learned projection layers act on the last state of the ASR decoder and are added as a residual connection to the LLM decoder layer's representations. The advantage of such a coupling is that the ASR's cross-attention layers are retained, and a simple projection from the ASR decoder states to the LLM decoder states suffices. We will show that such a coupling leads to significantly faster training of the coupling parameters than existing approaches such as Whispering-Llama [6].

SALSA can be used to integrate any pretrained decoder-only LLM with a pretrained encoder-decoder ASR model using small amounts of labeled speech in the target languages. By projecting just the last state of the ASR decoder at each LLM decoding step, we bypass the need for learning cross-attention modules, thus making SALSA significantly more parameter-efficient than existing approaches. We implement SALSA using a pretrained Whisper ASR model [10] and LLama-2 [4]. On eight diverse languages in the FLEURS [11] benchmark, we obtain a significant 16% on average and up to a maximum of 38% relative reduction in WER compared to parameter-efficient finetuning. In contrast, existing fusion approaches that rely on n-best lists from ASR perform much worse for low resource languages.

2. Related Work

LM adaptation in ASR. Traditional ASR systems consist of decoupled acoustic and language models [12] which enables easier adaptation to a target domain [13, 14, 15, 16, 17, 18]. For end-to-end ASR models, the most popular approach to

[*]These authors contributed equally to this work.

[1]Code for SALSA is available at https://github.com/csalt-research/salsa.

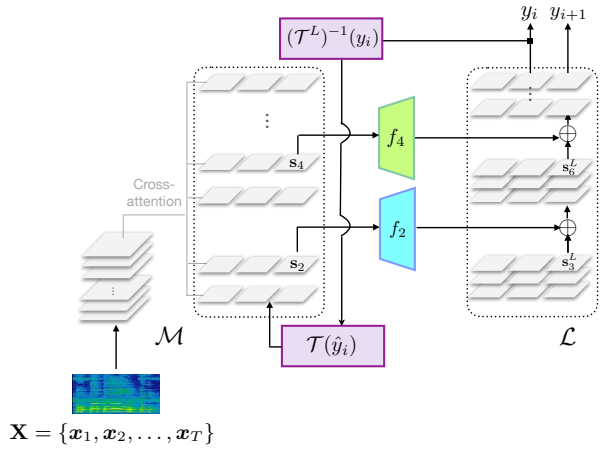

Figure 1: *SALSA: Overall schematic illustrating the coupling of the ASR model $\mathcal{M}$ and the LLM $\mathcal{L}$ using select projection layers. For the sake of simplicity, we will assume y_i corresponds to a single token in this illustration.*

integrate an external LM is shallow fusion and its variants, where an external LM is log-linearly interpolated with the ASR model [19, 20, 21, 22, 23, 24, 25].

LLM adaptation in ASR. With the rapid progress on various natural language tasks using LLMs, their integration with ASR models is emerging as an area of significant interest. One of the early application of LLMs was to use them for ASR error correction by providing an n-best list and a prompt to generate the correct prediction [7, 26, 27, 28]. These solutions are heavily reliant on the ASR outputs and will not fare well on low-resource languages owing to large errors in the n-best predictions.

Deep LLM integration with an ASR system. An active area of research is to integrate the audio modality within an LLM. Whispering-Llama [6] learns adapter layers [29, 30] to cross-attend to the audio features and prompts with an n-best list to improve English ASR. Another popular approach is to provide the output of an audio encoder directly as an input to decoder-only LLMs. Full fine-tuning is done to semantically map the acoustic features with the underlying textual features within an LLM [3, 5, 9, 31, 32, 33]. The closest to our work is [34], that does a late integration of the ASR and LLM decoders. However, they work with n-best lists (that is not conducive to low-resource languages) and finetune the ASR decoder using the LLM's tokenizer to match the LLM vocabulary. SALSA does not need such an additional finetuning step and can work with different ASR/LLM tokenizations.

3. Our Approach: SALSA

The input to our model is a trained ASR model $\mathcal{M}$, an LLM model $\mathcal{L}$, and a small set of labeled audio-transcript pairs $D = \{(\boldsymbol{x}^n, \boldsymbol{y}^n) : n = 1 \ldots N\}$ in a target low-resource language. We assume that the LLM has been pre-trained with significantly more text data in the target language, compared to the speech transcription data used in the ASR model. We first present a brief background of the ASR and LLM models.

Background: ASR model. We assume the ASR system $\mathcal{M}$ is an encoder-decoder model as used in state-of-the-art ASR systems like Whisper [10]. The encoder $\mathcal{M}_S$ converts an input audio X comprising of T frames $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$ into their latent vectors $\mathbf{h} = \mathbf{h}_1, \ldots, \mathbf{h}_T$. The transcript $\boldsymbol{y}$ is generated autoregressively as per the vocabulary $\mathcal{V}$ and a tokenization algorithm $\mathcal{T}$. Let $\boldsymbol{y}_u$=$y_1, \ldots, y_u$ denote the token sequence of the transcript generated so far. The decoder takes as input the prefix $\boldsymbol{y}_u$ and cross-attends on the audio states $\mathbf{h}_1, \ldots, \mathbf{h}_T$ to generate the next token via multiple self-attention layers. Let d denote the number of decoder layers. Let $\mathbf{s}_\ell \in \Re^m$ denote the vector output from the decoder at layer $\ell \in [1, d]$.

Background: LLM model. We assume the $\mathcal{L}$ is a decoder-only model consisting of d^L layers. Conditioned on an instruction prompt and the partially generated text, the LLM also generates the next token autoregressively. Let $\mathcal{V}^L$ denote the vocabulary of the LLM and let $\mathcal{T}^L$ denote its tokenizer. In general, both $\mathcal{V}^L$ and $\mathcal{T}^L$ could be different from $\mathcal{V}$ and $\mathcal{T}$ respectively of the ASR model. At any step t of generation, let $\mathbf{s}_\ell^L \in \Re^{m^L}$ denote the decoder output from each layer $\ell \in [1, d^L]$. Finally, a softmax yields $P(y|\mathbf{s}_{d^L}^L)$ where $y \in \mathcal{V}^L$.

Our method of coupling the ASR model $\mathcal{M}$ with the LLM model $\mathcal{L}$ is to just add a lightweight bridge from the latest state of the ASR decoder to the LLM decoder state as shown in Figure 1. We choose a subset $F \subseteq \{1 \ldots, d^L\}$ of LLM decoder layers to connect to $|F|$ different ASR decoder layers. Let $I(\ell)$ denote the index of the ASR decoder layer to which a LLM decoder layer $\ell \in F$ is connected. We use $f_\ell : \mathbb{R}^m \rightarrow \mathbb{R}^{m^L}$ to denote the feed forward network for the ℓth layer. The output of this function is used to add a residual connection to the LLM's decoder output at layer ℓ as:

$$\mathbf{s}_\ell^L = \mathbf{s}_\ell^L + f_\ell(\mathbf{s}_{I(\ell)}) \quad (1)$$

The set of parameters over all $f_\ell : \ell \in F$ is denoted as θ_C. We experimented with different methods of choosing the set F and I. Our default method, if size of F is k, is to make F every d^L/F^{th} layer of $\mathcal{L}$, and $I(F)$ be the corresponding d/F^{th} decoder layer of $\mathcal{M}$. We will present ablation on size of F, and alternative methods of choosing F.

Transcript generation. The LLM model $\mathcal{L}$ generates the next token auto-regressively. The LLM's softmax layer yields a distribution $P(y|\mathbf{s}_d^L)$ over its vocabulary $\mathcal{V}^L$. From this, the sampled y_{t+1} forms the next generated token. In general, this token may not be valid text recognized by the ASR tokenizer $\mathcal{T}$. Often for low resource languages the tokenizers for $\mathcal{L}$ and $\mathcal{M}$ can use different multi-token sequences to encode a character in the target language. So, the LLM keeps generating tokens until a valid text piece recognizable by ASR's $\mathcal{T}$ is formed. As soon as the LLM decodes a complete character sequence (which can be a single character or a set of characters), the just generated text is re-tokenized with the ASR's tokenizer to convert into a sequence of tokens that can be understood by $\mathcal{M}$. Finally, the decoder state of $\mathcal{M}$ is advanced with the newly predicted sequence of tokens. This updated decoder state is then used by $\mathcal{L}$ in its subsequent decoding iterations. The synchronous invocation of different decoders with different vocabularies is a distinctive aspect of our approach. Algorithm 1 gives an overview of decoding in SALSA.

Training. We train only the coupling parameters θ_C using the limited training dataset D in the target language. The parameters of both the ASR and LLM models are kept frozen. During training, an added detail is to tokenize a gold transcript $\boldsymbol{y}$ first using the LLM's tokenizer $\mathcal{T}^L$, then tokenize using the conditional tokenization of the ASR model as elaborated above, and finally remember for each index in the LLM's token se-

Algorithm 1: SALSA Decoding Algorithm

Input : ASR Model $\mathcal{M}$, LLM Model $\mathcal{L}$, Audio: X
Tokenizers $\mathcal{T}$ of ASR, $\mathcal{T}^L$ of LLM
f_ℓ, F, I : coupling model specification.
Output: Transcript $\{\boldsymbol{y}_0, \ldots, \boldsymbol{y}_u\}$

$\mathbf{h}$: ASR Encoder State (X)
$\mathbf{s}$ = ASR Decoder State initialized with SOT token
$\mathbf{s}^L$ = Initial LLM Decoder State
pi = 0, $\boldsymbol{y}$ = []
for $i \in \{1 \ldots$ *max estimated tokens in* $X\}$ **do**
  $\mathbf{s}_\ell^L = \mathbf{s}_\ell^L + f_\ell(\mathbf{s}_{I(\ell)}), \quad \forall \ell \in F$
  $y_i, \mathbf{s}^L = \mathcal{L}(\mathbf{s}^L)$
  $\boldsymbol{y}$.append(y_i)
  textY = $(\mathcal{T}^L)^{-1}(\boldsymbol{y})$
  if *textY ends with a valid utf-8 character* **then**
    new_tokens = $\mathcal{T}$(textY[pi:])
    $\mathbf{s} = \mathcal{M}(\mathbf{h}, \mathbf{s},$ new_tokens)
    pi = `len`(textY)
  If $y_i == \mathcal{T}^L$.eos break;
return $\boldsymbol{y}$

quence, the aligned sequence in the ASR tokenization. Otherwise, training proceeds using teacher forcing as in normal encoder-decoder models with cross entropy loss on the probabilities produced by LLM's decoder.

In spite of the need to handle such differences in tokenization, we found two advantages of such a coupling: (1) Since the ASR decoder has already been trained for cross-attention on the encoded audio, the LLM decoder does not need to be retrained for this task. Instead, since both the ASR and LLM decoders are auto-regressive, the LLM only needs to consult the last ASR decoder state. (2) Since the LLM is assumed to be better at modeling the target language, the final text generation is with the LLM's decoder with a residual connection to the ASR decoder.

4. Experimental Setup

Models. We use the Whisper Large-v2 encoder-decoder model for ASR (1.55B parameters). For the LLM, we experiment with two models of varying size, namely LLaMA2-7B and LLaMA2-13B. SALSA additionally uses 8 projection layers that each first reduce the dimensionality down from 768 (matching Whisper Large-v2's dimensionality) to 192, use SiLU activations, and then project up from 192 to match the dimensionality of the corresponding LLaMA2 model (i.e 4096 for LLaMA2-7B and 5120 for LLaMA2-13B). For all our SALSA experiments, we use Lit-GPT [36] library to modify the LLM decoding to work in tandem with the ASR decoder.

Dataset. We evaluate on a subset of the FLEURS multilingual benchmark dataset [11]. This is an n-way parallel dataset that consists of roughly 12 hours of supervision for over 100 languages. Our conjecture is that SALSA could be beneficial in improving the ASR performance of those low-resource languages for which Whisper's ASR performance is poor, incurring moderate to high word error rates (WERs), and for which the LLM has reasonable text generation capabilities. The latter is quantitatively measured using the following metric. For a validation set of roughly 500 sentences in each evaluation language, we use the LLM to autoregressively predict only the next character given the past history. We loosely treat a single character across languages as having roughly similar durations in its underlying speech. For each evaluation language, we measure the fraction of next character predictions (NCP-acc) that exactly match the ground-truth characters. Figure 2 shows both Whisper's WERs and NCP-acc scores for a subset of FLEUR languages. We are interested in languages that appear in the upper and lower right quadrants. That is, those languages that incur moderate to high WERs and have fairly high NCP-acc scores. Based on this analysis, our final set of evaluation languages are Hindi, Gujarati, Marathi, Malayalam, Persian, Punjabi, Tamil and Telugu. We do not choose Burmese because severe under-tokenization results in most utterances exceeding Whisper decoder's fixed length of 448 tokens. We do not choose Vietnamese as the WER with pretrained Whisper is already quite good (10.3%).

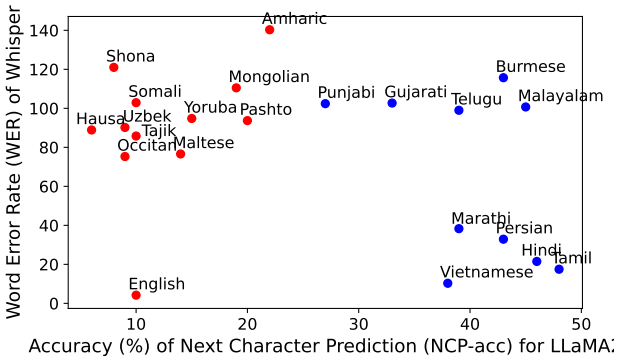

Figure 2: *2D plot comparing the accuracy (%) of LLaMA2 on Next Character Prediction (NCP-acc) with the Word Error Rate (WER %) of Whisper on a subset of languages from FLEURS. The plot serves as a point of reference for selecting languages that might benefit from SALSA. Specifically, we chose languages (colored in blue) that have high NCP-acc and medium to high WER using Whisper.*

Training and Inference. For all our chosen evaluation languages, we train the model for 35 epochs with a batch size of 32, a learning rate of 0.001 and a maximum of 2000 steps. We use the AdamW [37] optimizer with a weight decay of 0.02. To alleviate hallucinations during inference, we use nucleus sampling with top-p value of 0.9 and top-k value of 10. For some utterances on which the ASR performs very poorly, the LLM is prone to repetitions and gets stuck in a loop predicting up to $\max_l$ tokens without predicting the end-of-sentence token. To mitigate this issue, we train a simple duration-to-length regressor for each language that takes the duration of the speech as its input and predicts an estimate of the number of output tokens. If the length of the LLM prediction far exceeds this predicted length, we truncate the overall prediction to the estimated length from the regressor.

Baselines. We compare SALSA with Whisper-v2 finetuned on the labeled data in each language using LoRA fine-tuning [35]. We also compare against a recent ASR-LLM fusion model Whispering-Llama [6] that prompts the LLM with n-best lists from the Whisper ASR and trains the LLM via cross-attention adapter modules to attend to the Whisper encoder states.

Table 1: *Comparison of WERs (%) of SALSA using LLaMA-7B and LLaMA-13B models against Whispering-LlaMA and different variants of Whisper model on eight languages of* FLEURS. *Numbers in bold without denote the best across baselines and with denotes the best WER across all experiments.* † *indicates that the performance gains are statistically significant at* $p < 0.001$.

Method	# params	Gujarati	Hindi	Malayalam	Marathi	Persian	Punjabi	Tamil	Telugu	Average
Our Baselines (Reproduced using official repositories)										
Whisper (Large-v2) [10]	–	108.2	35.9	107.8	84.7	35.8	101.8	48.0	104.4	78.3
w/ LoRA fine-tuning [35]	15M	**55.7**	**19.3**	**54.9**	**35.8**	**16.9** †	**46.7**	**38.0** †	**47.5**	**39.4**
Whispering-LlaMA [6]	26M	90.8	53.2	101.1	105.9	86.1	92.6	89.1	106.4	90.7
SALSA-7B (w/ LLaMA2-7B)										
w/ Whisper (Large-v2)	17M	37.8	18.2	40.9	37.5	18.6	37.0	40.1	44.9	34.4
w/ LoRA fine-tuned Whisper	17M	**34.6** †	17.6	35.3	**35.6** †	18.2	34.8	42.6	45.1	33.0
SALSA-13B (w/ LLaMA2-13B)										
w/ Whisper (Large-v2)	19M	37.1	17.4	40.4	37.8	18.6	37.0	40.1	45.0	34.1
w/ LoRA fine-tuned Whisper	19M	34.9	**16.8** †	**34.8** †	36.5	17.6	**34.5** †	41.4	**45.0** †	**32.7** †

Table 2: *Comparison of the performance (WER %) of* ***multilingual*** *SALSA using LLaMA2-7B.*

Method	# params	Gujarati	Hindi	Malayalam	Marathi	Punjabi	Tamil	Telugu	Average
Whisper (Large-v2) [10]	–	108.2	35.9	107.8	84.7	101.8	48.0	104.4	84.4
w/ LoRA fine-tuning [35]	15M	56.8	19.8	**39.8** †	37.4	55.8	**39.8** †	50.5	42.8
SALSA-7B w/ Whisper (Large-v2)	17M	**36.4** †	**17.5** †	40.0	**36.7** †	**36.1** †	40.2	**43.2** †	**35.8** †

5. Experimental Results and Analysis

Table 1 shows the overall ASR results across eight languages comparing SALSA with the two baseline systems. Whispering-Llama underperforms due to the poor quality of the n-best lists derived from Whisper for these low-resource languages. SALSA can be used either with a pretrained or a finetuned Whisper ASR model. We are careful to train the same number of parameters using SALSA and LoRA finetuning. We observe that SALSA on top of a finetuned Whisper significantly outperforms the finetuned Whisper baseline by an overall relative WER reduction of 16%. This attests to SALSA's ability to derive complementary benefits over and above finetuning. The bigger Llama-13B model does not offer any consistent advantage over Llama-7B; the latter yields the best WERs for four test languages.

Multilingual SALSA. We test how SALSA performs with multilingual data used to train projection layers shared across all languages. We pick seven of the Indian languages for this experiment. As shown in Table 2, SALSA outperforms the multilingual fine-tuned Whisper model by 20% signifying the superior adaptation capability of SALSA with similar number of parameters.

Ablation Analysis. We study the effect of the number of adaptation parameters and the positions of adaptation layers on SALSA, as shown in Table 3. We observe that using 8 adapter layers all at the end of the decoder yields almost similar performance as the model which has adaptation layers uniformly distributed. Reducing the number of adaptation layers for both SALSA-7B and SALSA-13B results in a degradation of around relative 5% WER, but is still significantly better than LoRA finetuning which yields an average WER of 39.4%. This shows that SALSA can adapt and generalize much better than existing baselines even with a small number of parameters.

Runtime Complexity. We compare the real-time factor (RTF) of SALSA with the original Whisper model and Whispering-Llama model. That is, the amount of time it takes to decode 1 sec of audio. RTFs of Whisper, SALSA and Whispering-Llama on an A100 80GB GPU are 0.42 secs, 0.64 secs and 1.3 secs, respectively. For the Whispering-Llama model, a significant fraction of the total time ($> 60\%$) is spent in generating n-best lists which adversely affects the overall RTF. For training, SALSA takes around 1 hour on an A100 80GB GPU, while Whispering-Llama takes around 6 hours owing to high training overhead in generating n-best lists and learning expensive cross-attention over significantly long audio embeddings. LoRA finetuning takes roughly the same time as Whispering-Llama on a smaller A100 40GB GPU (as there are no Llama2 weights to load). SALSA is much faster to train in comparison to the baselines owing to its simple architecture.

Table 3: *Ablation study comparing the number of adapter layers and the position of adapter layers averaged across 8 languages.*

Method	# params	Average WER
SALSA-7B (w/ LLaMA2-7B)		
8 adapter layers (uniformly distributed)	17M	34.4
8 adapter layers (all at the end)	17M	34.8
4 adapter layers (uniformly distributed)	8.5M	36.4
4 adapter layers (all at the end)	8.5M	36.9

6. Conclusion

In this paper we presented SALSA, a light-weight fusion of an ASR system with an LLM that retains the ASR model's expertise in encoding and decoding audio, while harnessing the superior language modeling capabilities of the LLM. Our method provides significant reductions in WER compared to fine-tuning the ASR model alone, while providing efficient one-pass decoding, and much faster training than existing LLM-ASR fusion methods. In this paper, we focused on improving the transcription of isolated utterances. In future, we plan to harness the instruction-following capabilities of LLMs for more applications that require stateful contextual biasing.

7. References

[1] P. Xu and P. Fung, "Cross-lingual language modeling for low-resource speech recognition," *IEEE transactions on audio, speech, and language processing*, vol. 21, no. 6, pp. 1134–1144, 2013.

[2] A. Gandhe, F. Metze, and I. Lane, "Neural network language models for low resource languages," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[3] J. Wu, Y. Gaur, Z. Chen, L. Zhou, Y. Zhu, T. Wang, J. Li, S. Liu, B. Ren, L. Liu *et al.*, "On decoder-only architecture for speech-to-text and large language model integration," in *Proceedings of ASRU*. IEEE, 2023, pp. 1–8.

[4] H. Touvron *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[5] Y. Fathullah, C. Wu, E. Lakomkin, J. Jia, Y. Shangguan, K. Li, J. Guo, W. Xiong, J. Mahadeokar, O. Kalinli *et al.*, "Prompting large language models with speech recognition abilities," *arXiv preprint arXiv:2307.11795*, 2023.

[6] S. Radhakrishnan *et al.*, "Whispering llama: A cross-modal generative error correction framework for speech recognition," in *Proceedings of EMNLP*, 2023, pp. 10 007–10 016.

[7] C. Chen, Y. Hu, C.-H. H. Yang, S. M. Siniscalchi, P.-Y. Chen, and E.-S. Chng, "Hyporadise: An open baseline for generative speech recognition with large language models," *Proceedings of NeurIPS*, vol. 36, 2024.

[8] D. Zhang, S. Li, X. Zhang, J. Zhan, P. Wang, Y. Zhou, and X. Qiu, "Speechgpt: Empowering large language models with intrinsic cross-modal conversational abilities," *arXiv preprint arXiv:2305.11000*, 2023.

[9] J. Pan, J. Wu, Y. Gaur, S. Sivasankaran, Z. Chen, S. Liu, and J. Li, "Cosmic: Data efficient instruction-tuning for speech in-context learning," *arXiv preprint arXiv:2311.02248*, 2023.

[10] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," in *Proceedings of ICML*, ser. ICML'23. JMLR.org, 2023.

[11] A. Conneau *et al.*, "Fleurs: Few-shot learning evaluation of universal representations of speech," in *Proceedings of SLT*. IEEE, 2023, pp. 798–805.

[12] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 69–88, 2002.

[13] T. Hori, D. Willett, and Y. Minami, "Language model adaptation using wfst-based speaking-style translation," in *Proceedings of ICASSP*, vol. 1. IEEE, 2003, pp. I–I.

[14] J. R. Bellegarda, "Statistical language model adaptation: review and perspectives," *Speech communication*, vol. 42, no. 1, pp. 93–108, 2004.

[15] G. Neubig, S. Mori, and T. Kawahara, "A wfst-based log-linear framework for speaking-style transformation," in *Tenth Annual Conference of the International Speech Communication Association*. Citeseer, 2009.

[16] S. R. Gangireddy, P. Swietojanski, P. Bell, and S. Renals, "Unsupervised adaptation of recurrent neural network language models." in *Proceedings of Interspeech*, 2016, pp. 2333–2337.

[17] J. Park, X. Liu, M. J. Gales, and P. C. Woodland, "Improved neural network based language modelling and adaptation," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[18] H. Xu, T. Chen, D. Gao, Y. Wang, K. Li, N. Goel, Y. Carmiel, D. Povey, and S. Khudanpur, "A pruned rnnlm lattice-rescoring algorithm for automatic speech recognition," in *Proceedings of ICASSP*. IEEE, 2018, pp. 5929–5933.

[19] A. Kannan, Y. Wu, P. Nguyen, T. N. Sainath, Z. Chen, and R. Prabhavalkar, "An analysis of incorporating an external language model into a sequence-to-sequence model," in *Proceedings of ICASSP*, 2018, pp. 1–5828.

[20] E. McDermott, H. Sak, and E. Variani, "A density ratio approach to language model fusion in end-to-end automatic speech recognition," in *Proceedings of ASRU*. IEEE, 2019, pp. 434–441.

[21] Z. Meng, S. Parthasarathy, E. Sun, Y. Gaur, N. Kanda, L. Lu, X. Chen, R. Zhao, J. Li, and Y. Gong, "Internal language model estimation for domain-adaptive end-to-end speech recognition," in *Proceedings of SLT*. IEEE, 2021, pp. 243–250.

[22] Z. Meng, Y. Wu, N. Kanda, L. Lu, X. Chen, G. Ye, E. Sun, J. Li, and Y. Gong, "Minimum Word Error Rate Training with Language Model Fusion for End-to-End Speech Recognition," in *Proceedings of Interspeech*, 2021, pp. 2596–2600.

[23] T. Udagawa, M. Suzuki, G. Kurata, N. Itoh, and G. Saon, "Effect and analysis of large-scale language model rescoring on competitive asr systems," *arXiv preprint arXiv:2204.00212*, 2022.

[24] A. Mittal, S. Sarawagi, and P. Jyothi, "In-situ text-only adaptation of speech models with low-overhead speech imputations," in *Proceedings of ICLR*, 2023.

[25] A. Mittal, S. Sarawagi, P. Jyothi, G. Saon, and G. Kurata, "Speech-enriched memory for inference-time adaptation of asr models to word dictionaries," in *Proceedings of EMNLP*, 2023.

[26] P. Dighe, Y. Su, S. Zheng, Y. Liu, V. Garg, X. Niu, and A. Tewfik, "Leveraging large language models for exploiting asr uncertainty," *arXiv preprint arXiv:2309.04842*, 2023.

[27] C.-H. H. Yang, Y. Gu, Y.-C. Liu, S. Ghosh, I. Bulyko, and A. Stolcke, "Generative speech recognition error correction with large language models and task-activating prompting," in *Proceedings of ASRU*. IEEE, 2023, pp. 1–8.

[28] R. Ma, M. Qian, P. Manakul, M. Gales, and K. Knill, "Can generative large language models perform asr error correction?" *arXiv preprint arXiv:2307.04172*, 2023.

[29] R. Zhang, J. Han, A. Zhou, X. Hu, S. Yan, P. Lu, H. Li, P. Gao, and Y. Qiao, "Llama-adapter: Efficient fine-tuning of language models with zero-init attention," *arXiv preprint arXiv:2303.16199*, 2023.

[30] P. Gao, J. Han, R. Zhang, Z. Lin, S. Geng, A. Zhou, W. Zhang, P. Lu, C. He, X. Yue *et al.*, "Llama-adapter v2: Parameter-efficient visual instruction model," *arXiv preprint arXiv:2304.15010*, 2023.

[31] C.-I. J. Lai, Z. Lu, L. Cao, and R. Pang, "Instruction-following speech recognition," *arXiv preprint arXiv:2309.09843*, 2023.

[32] Y. Shu, S. Dong, G. Chen, W. Huang, R. Zhang, D. Shi, Q. Xiang, and Y. Shi, "Llasm: Large language and speech model," *arXiv preprint arXiv:2308.15930*, 2023.

[33] Z. Ma, G. Yang, Y. Yang, Z. Gao, J. Wang, Z. Du, F. Yu, Q. Chen, S. Zheng, S. Zhang *et al.*, "An embarrassingly simple approach for llm with strong asr capacity," *arXiv preprint arXiv:2402.08846*, 2024.

[34] C. Chen, R. Li, Y. Hu, S. M. Siniscalchi, P.-Y. Chen, E. Chng, and C.-H. H. Yang, "It's never too late: Fusing acoustic information into large language models for automatic speech recognition," *arXiv preprint arXiv:2402.05457*, 2024.

[35] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in *Proceedings of ICLR*, 2022. [Online]. Available: https://openreview.net/forum?id=nZeVKeeFYf9

[36] L. AI, "Lit-gpt," https://github.com/Lightning-AI/lit-gpt, 2023.

[37] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.