ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 979-8-3503-6874-1/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICASSP49660.2025.10890391

Delayed Fusion: Integrating Large Language Models into First-Pass Decoding in End-to-end Speech Recognition

Takaaki Hori[1], *Martin Kocour*[2*], *Adnan Haider*[1], *Erik McDermott*[1], *Xiaodan Zhuang*[1]

[1]Apple, [2]Brno University of Technology

***Abstract*—This paper presents an efficient decoding approach for end-to-end automatic speech recognition (E2E-ASR) with large language models (LLMs). Although shallow fusion is the most common approach to incorporate language models into E2E-ASR decoding, we face two practical problems with LLMs. (1) LLM inference is computationally costly. (2) There may be a vocabulary mismatch between the ASR model and the LLM. To resolve this mismatch, we need to retrain the ASR model and/or the LLM, which is at best time-consuming and in many cases not feasible. We propose *delayed fusion*, which applies LLM scores to ASR hypotheses with a delay during decoding and enables easier use of pre-trained LLMs in ASR tasks. This method can reduce not only the number of hypotheses scored by the LLM but also the number of LLM inference calls. It also allows re-tokenizion of ASR hypotheses during decoding if ASR and LLM employ different tokenizations. We demonstrate that delayed fusion provides improved decoding speed and accuracy compared to shallow fusion and N-best rescoring using the LibriHeavy ASR corpus and three public LLMs, OpenLLaMA 3B & 7B and Mistral 7B.**

***Index Terms*—speech recognition, large language model, decoding, delayed fusion**

I. Introduction

Large language models (LLMs) have shown their tremendous power of language understanding and generation in various domains [1]–[4]. LLMs, including many publicly available ones [5], are typically Transformer models [6] with billions of parameters trained on vast amounts of text data. Towards effectively exploiting LLMs for ASR, researchers are very actively exploring various LLM-based ASR models and decoding approaches [7]–[9].

Most ASR systems employ an external language model to improve recognition accuracy. If similarly applying an LLM using conventional shallow fusion, we face two practical problems. (1) LLM inference is computationally demanding, making it costly to directly apply shallow fusion during beam search, which requires many LLM inference calls, especially in frame-synchronous decoding. (2) There may be a vocabulary mismatch between ASR model and LLM. LLMs typically have a much larger vocabulary compared to end-to-end ASR models. To apply shallow fusion, ASR model and LLM need to have identical vocabularies. To match one vocabulary to another, either the ASR model or the LLM needs to be retrained. However, training an ASR model on the LLM vocabulary leads to an out-of-vocabulary problem, since the paired data used for training ASR models is limited and does not cover LLM vocabulary sufficiently well. On the other hand, training an LLM on the ASR vocabulary is expensive, time consuming and in many cases infeasible. Furthermore, publicly available pre-trained LLMs cannot be easily adapted to different vocabularies.

N-best rescoring is a possible solution to relieve the above problems: first-pass decoding generates N-best hypotheses, and then a second-pass rescores the hypotheses using an LLM, after re-tokenization. The rescoring pass is not very expensive if a graphics processing unit (GPU) is available, since rescoring requires only one LLM inference call when first-pass hypotheses are batched. However, N needs to be large enough to make rescoring effective. Especially for long utterances, it is difficult to generate short N-best lists that include the correct hypothesis. Increasing the list size imposes a big burden on the first-pass decoding, where more computation and memory are needed for a wider beam, especially when using an auto-regressive E2E model that predicts the next tokens based on all the previous tokens.

We propose *delayed fusion*, where we apply LLM scores during decoding but only after pruning, which dramatically reduces the number of partial hypotheses that need to be scored by the LLM. At the same time, we can wait until a partial hypothesis reaches the end of word to handle different tokenizations between ASR and LLM. Once the decoder detects the end of a word, it re-tokenizes the word, computes the LLM score and adds it to the current partial hypothesis score. This way, LLM scores are incorporated from an early stage of the first-pass decoding, reducing search errors otherwise not recoverable by N-best rescoring.

The contributions of this work are:

- We propose a novel method for efficient LM fusion, which allows us to (1) easily compare different LLMs on ASR tasks, (2) investigate the effect of prompting LLMs in ASR, and (3) use as a baseline system when exploring advanced LLM-based ASR models.
- We provide experimental results on ASR accuracy and decoding speed with three public domain LLMs, OpenLLaMA 3B & 7B [10], [11] and Mistral 7B [4], showing that (1) Delayed LLM fusion is fast enough compared to standard neural language model (NLM) fusion, allowing us to obtain improved ASR accuracy from LLMs in the same decoding time, and that (2) Delayed LLM fusion provides significant WER reduction compared to N-best rescoring with LLMs.

II. Related work

There are different types of E2E-ASR systems [12] and many of them employ an external LM to improve recognition accuracy [13]–[15]. Some LM fusion techniques require retraining of the ASR model to further improve accuracy and adaptability to other domains [16]–[18]. Unlike such techniques, this paper focuses on improving shallow fusion based decoding, which combines E2E-ASR and LM without retraining or fine-tuning.

In standard shallow fusion decoding, ASR model and LM need to use the same tokenization and vocabulary. This limitation prevents us from easily applying LLM shallow fusion. Prior work has investigated delayed LM application using on-the-fly lattice rescoring [19], [20] for hybrid ASR and shallow fusion of a character-based E2E model and a word-based LM for end-to-end ASR, where a space token is used to trigger word-based LM scoring [21], [22]. However, these approaches do not have a mechanism to control the delay for efficient shallow fusion. With delayed fusion, we can adjust the timing of LM

[*]This work was done during an internship at Apple.

fusion considering the computation vs. accuracy trade-off as well as tokenization mismatches. Moreover, unlike LLM rescoring [23]–[26] or redecoding [27], [28] approaches, delayed fusion can be used in streaming scenarios, similarly to standard shallow fusion.

A recent work related to our approach is SALSA [9] since it can handle tokenization mismatches during decoding. However, SALSA integrates pre-trained ASR and LLM by combining their state vectors using additional projection layers, and thus, it needs to train those layers using paired data before decoding. On the other hand, delayed fusion integrates ASR and LLM by combining their scores, and therefore, it does not require any extra layers or training steps.

III. Method

A. Delayed fusion concept

Delayed fusion computes LM scores for partial hypotheses during decoding as in shallow fusion. However, such scoring is done after pruning, enabling flexible timing of fusion balancing the trade-off between accuracy and computational cost. General auto-regressive decoding with delayed fusion is described in Algorithm 1.

Algorithm 1 Auto-regressive decoding with delayed fusion

```
 1: $H_0 \leftarrow \{\texttt{<s>}\}$
 2: $S_{E2E}(\texttt{<s>}) \leftarrow S_{LM}(\texttt{<s>}) \leftarrow 0$
 3: for $t = 1, 2, \ldots, T$ do
 4:     $H_t \leftarrow \textsc{Extend}(H_{t-1})$
 5:     $S_{E2E} \leftarrow \text{E2E}\textsc{Score}(H_t, S_{E2E})$
 6:     $H_t \leftarrow \textsc{Prune}(H_t, S_{E2E}, S_{LM}, K)$
 7:     if $\textsc{Fusable}(H_{0:t}, t) = \textsc{True}$ then,
 8:         $S_{LM} \leftarrow \text{LM}\textsc{Score}(H_t, S_{LM})$
 9:     end if
10: end for
11: $\hat{H}_T \leftarrow \textsc{Finalize}(H_T)$
12: $S_{LM} \leftarrow \text{LM}\textsc{Score}(\hat{H}_T, S_{LM})$
13: $\hat{h} \leftarrow \text{argmax}_{h \in \hat{H}_T}(S_{E2E}(h) + S_{LM}(h))$
```

The method first creates an initial hypothesis list with begin-of-sentence token <s> (line 1) and initializes E2E model score list $S_{E2E}(\texttt{<s>})$ and LM score list $S_{LM}(\texttt{<s>})$ (line 2). For each step t, it extends the previous hypothesis list H_{t-1} to get the current hypothesis list H_t, and computes E2E model scores $S_{E2E}(h)$ for $h \in H_t$ (lines 4–5). Then, it applies pruning for H_t to keep the top K hypotheses based on S_{E2E} and S_{LM} (line 6). If the fusion condition $\textsc{Fusable}(H_{0:t}, t)$ is met,it computes LM scores $S_{LM}(h)$ for $h \in H_t$ (lines 7–9). After T steps, it updates the hypothesis list H_T, appending the end-of-sentence token </s> to each h in H_T if necessary (line 11), and also computes LM scores S_{LM} for H_T (line 12). Finally, it selects the best hypothesis $\hat{h}$ based on S_{E2E} and S_{LM} (line 13).

Algorithm 1 shows the abstract-level decoding steps. In frame-synchronous decoding, t represents a time frame. For CTC decoding, $S_{E2E}(h)$ must have two elements for alignment paths ending with blank and non-blank labels respectively, which are updated according to the CTC rule [29] in $\text{E2E}\textsc{Score}(\cdot)$. In label-synchronous decoding, t represents the number of labels generated for each hypothesis, where all existing hypotheses have the same length. In addition, the algorithm requires a function that decides whether to exit the for loop or not, for example checking whether all existing hypotheses end with </s>. This step is omitted from the algorithm for simplicity.

Thus, the algorithm is agnostic to the decoding strategy, be it frame-synchronous or label-synchronous decoding, as long as it is auto-regressive. The key step is the LM score computation in line 8, which is performed after pruning. The timing can be controlled by the function $\textsc{Fusable}(\cdot)$ to reduce the number of LM calls for efficiency.

B. Delayed fusion with LLM

If the ASR model and the LLM were trained with different vocabularies, we need to re-tokenize hypotheses before LLM scoring. However, tokenization may be incorrect for incomplete hypotheses. Accordingly, we determine a tokenizable sub-sequence as the longest prefix that ends with a word-end token occurring right before a word-begin token, as shown in Fig. 1. Then, the sequence is re-tokenized using the LLM tokenizer. If a standard SentencePiece tokenizer [30] is used, each word is tokenized into a unique token sequence. Therefore, the sequence can be re-tokenized into a unique and consistent token sequence, for which an LLM score can be computed correctly.

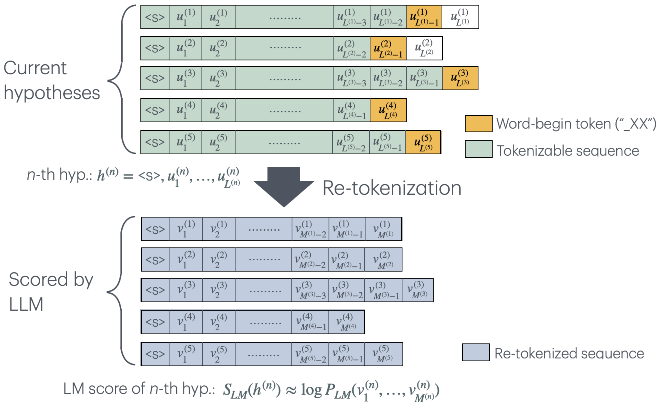
Current hypotheses
<s>
Word-begin token ("_XX")
Tokenizable sequence
n-th hyp.: $h^{(n)} = \texttt{<s>}, u_1^{(n)}, \ldots, u_{L^{(n)}}^{(n)}$
Re-tokenization
Scored by LLM
Re-tokenized sequence
LM score of n-th hyp.: $S_{LM}(h^{(n)}) \approx \log P_{LM}(v_1^{(n)}, \ldots, v_{M^{(n)}}^{(n)})$

Fig. 1. Re-tokenization for delayed LLM fusion.

With delayed fusion, we can call the LLM at any time during decoding. For efficient LLM computation, we propose (1) shortest-hypothesis fusion and (2) fixed-interval fusion. The shortest-hypothesis fusion calls the LLM only when the length of the shortest re-tokenized sequence has increased. The fusion condition is defined as

$$\textsc{Fusable}(H_{0:t}, t) = \begin{cases} \textsc{True} & \text{if } \varphi(\bar{H}_{t-1}) < \varphi(\bar{H}_t) \\ \textsc{False} & \text{otherwise} \end{cases},$$

where $\bar{H}_t$ is a list of re-tokenized sequences obtained from H_t or identical to H_t if there is no tokenization mismatch. $\varphi(\bar{H}_t)$ returns the length of the shortest sequence in $\bar{H}_t$. This method allows us to keep the number of LLM calls to at most the number of tokens in the shortest hypothesis.

The fixed-interval fusion calls the LLM at a fixed frame (or label) interval using

$$\textsc{Fusable}(H_{0:t}, t) = \begin{cases} \textsc{True} & \text{if } \bar{H}_{t-I} \neq \bar{H}_t \wedge t \bmod I = 0 \\ \textsc{False} & \text{otherwise} \end{cases},$$

where I denotes a pre-defined fixed interval. Increasing I reduces the number of LLM calls. Consequently, a larger I improves the decoding speed but increases the fusion delay, which may cause accuracy degradation. At the extreme, if $\textsc{Fusable}(\cdot)$ always returns $\textsc{False}$, the algorithm is equivalent to N-best rescoring, which in turn can be seen as a special case of delayed fusion.

For efficient LLM fusion, we compute LLM scores for all the current hypotheses at once, where we use hypothesis batching and

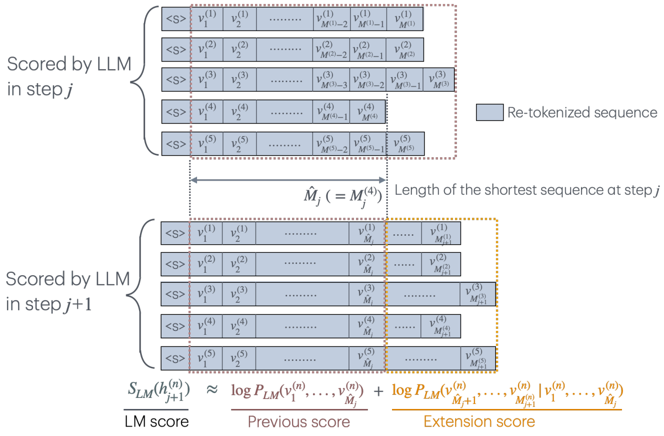
Scored by LLM in step j
<S>
$v_1^{(1)}$ $v_2^{(1)}$ .... $v_{M^{(1)}-2}^{(1)}$ $v_{M^{(1)}-1}^{(1)}$ $v_{M^{(1)}}^{(1)}$
Re-tokenized sequence
$\hat{M}_j\ (=M_j^{(4)})$
Length of the shortest sequence at step j
Scored by LLM in step $j+1$
$v_{\hat{M}_j}^{(1)}$ ...... $v_{M_{j+1}^{(1)}}^{(1)}$
$S_{LM}(h_{j+1}^{(n)}) \approx \log P_{LM}(v_1^{(n)},\ldots,v_{\hat{M}_j}^{(n)}) + \log P_{LM}(v_{\hat{M}_j+1}^{(n)},\ldots,v_{M_{j+1}^{(n)}}^{(n)} | v_1^{(n)},\ldots,v_{\hat{M}_j}^{(n)})$
LM score
Previous score
Extension score

Fig. 2. LLM score computation in decoding.

a key-value cache to take full advantage of GPU acceleration. Fig. 2 shows how to compute LLM scores at the $(j+1)$-th LLM call. The LM scores have already been computed for at least $\hat{M}_j$ tokens in the previous step j, where $\hat{M}_j$ denotes the length of the shortest sequence at step j. This means that we can use the key-value cache and the LLM scores for the preceding sequence $v_1^{(n)},\ldots,v_{\hat{M}_j}^{(n)}$ in computing new scores for all hypotheses. The new LLM scores at step $j+1$ can be obtained with

$$S_{LM}(h_{j+1}^{(n)}) \approx \log P_{LM}(v_1^{(n)},...,v_{\hat{M}_j}^{(n)}) + \log P_{LM}(v_{\hat{M}_j+1}^{(n)},...,v_{M_{j+1}^{(n)}}^{(n)} | v_1^{(n)},...,v_{\hat{M}_j}^{(n)}).$$

IV. Experiments

A. Conditions

We conducted several experiments on the LibriHeavy corpus [31], which includes 50k hours of English audio books. We trained a CTC-AED model [32], [33] using all three training subsets, i.e., small, medium, and large subsets, and also trained an in-domain NLM using formatted transcripts including casing and punctuation. The CTC-AED model had an encoder network with a Conv2D module followed by 12 Conformer blocks, a decoder network with 3 unidirectional Transformer blocks, and a CTC output layer. In the Conv2D module, 80-dimensional Mel-filter bank features obtained every 10 msec were down-sampled by a factor of 6. We employed multi-head attention of 8 heads with 512 dimensions in total. The feed-forward network had one hidden layer of 2,048 units with ReLU activations. The in-domain NLM had 9 Transformer blocks with 256 dimensions and a shared embedding layer for input and output tokens. The vocabulary size was 6K for both CTC-AED model and NLM, corresponding to the set of word pieces obtained from the LibriHeavy transcripts using the SentencePiece tokenizer [30]. The number of parameters of the CTC-AED model and the NLM were 101M and 9M, respectively. We applied SpecAugment in model training but did not use speed perturbation as in [31].

We also employed three public domain LLMs: OpenLLaMA 3B v2, OpenLLaMA 7B v2 [11], and Mistral 7B v0.1 [4], where the vocabulary size was 32K. We evaluated the proposed method in two decoding modes, CTC prefix beam search [21] and joint CTC-attention decoding [32], [34], where the former is frame-synchronous decoding and the latter is label-synchronous decoding. Although the frame-synchronous decoding is streamable, we used full utterance context in encoding to simplify the experiments. We used language model weights in shallow and delayed fusion, which were tuned on the LibriHeavy dev set for each LM. Evaluation metrics are word error rate (WER) and real-time factor (RTF). Decoding time was measured on an Intel Xeon (Skylake IBRS) CPU @ 2.4GHz with an NVIDIA V100 GPU.

TABLE I

Word error rate vs. real-time factor on LibriHeavy benchmarks[1]. Delayed fusion (DF) was used for the LLMs, while shallow fusion (SF) was used for the in-domain NLM.

Decoding	LM fusion	lh-clean	lh-other	RTF
CTC prefix beam search	-	3.47	6.41	0.018
	SF w/ NLM 9M	3.15	5.95	0.082
	DF w/ NLM 9M	3.16	5.93	0.066
	DF w/ OpenLLaMA 3B	3.05	5.68	0.115
	DF w/ OpenLLaMA 7B	**3.01**	5.67	0.141
	DF w/ Mistral 7B	3.02	**5.63**	0.142
CTC-attention decoding	-	2.98	5.64	0.051
	SF w/ NLM 9M	2.94	5.61	0.091
	DF w/ NLM 9M	2.96	5.59	0.099
	DF w/ OpenLLaMA 3B	2.86	5.35	0.146
	DF w/ OpenLLaMA 7B	2.84	5.29	0.170
	DF w/ Mistral 7B	**2.80**	**5.22**	0.169

B. Results

Table I shows WERs and RTFs for LibriHeavy test-clean ("lh-clean") and test-other ("lh-other") sets in different decoding conditions, where we compare the decoding modes, shallow fusion (SF) and delayed fusion (DF), and in-domain and large LMs, setting the beam size to 10. To measure the RTF, we used only the first 200 utterances in "lh-other". CTC prefix beam search without LM fusion is the baseline, which has the best RTFs, but high WERs. The WERs can be reduced by shallow fusion with the in-domain NLM, but RTF nonetheless increases significantly, even with the NLM probabilities computed on GPU. This is mainly due to frequent NLM calls during frame-synchronous decoding. We then evaluated delayed fusion using the shortest hypothesis approach described earlier. With the in-domain NLM, with no tokenization mismatch, delayed fusion shows comparable WERs with shallow fusion, while reducing RTF from 0.082 to 0.066. For the three LLMs, with the tokenization mismatch handled during decoding, delayed fusion produces better WERs than the baseline or NLM shallow fusion. Although the RTF increases for the LLMs to 0.115, 0.141, and 0.142, these are still acceptable for real-time decoding. With CTC-attention decoding, we see substantial improvements thanks to LLM fusion, although it does require more computation due to the use of the attention decoder and label-synchronous CTC. Note that, in label-synchronous decoding, delayed fusion based on the shortest hypothesis does not outperform shallow fusion in decoding speed (see SF vs. DF w/ NLM) because it does not reduce the number of LM calls. However, delayed fusion still has the benefit of handling mismatched tokenization. Since the performance gap between LLMs is small, we report only the results with OpenLLaMA 3B model in the following.

Table II compares delayed fusion strategies, the shortest- hypothesis and fixed-interval methods, with N-best rescoring. Both strategies achieve better WERs than the baseline and N-best rescoring. Moreover, by changing the interval I, we can choose a suitable

[1] The WERs presented in this paper cannot strictly be compared with those in [31] since our CTC-AED model is not compatible with their model in terms of vocabulary size, decoder architecture, and data augmentation.

TABLE II

COMPARISON OF DELAYED FUSION APPROACHES AND N-BEST RESCORING IN CTC PREFIX BEAM SEARCH. FOR FIXED-INTERVAL FUSION, WE TESTED INTERVALS OF $I = 16, 32$, AND 64, WHICH CORRESPOND TO 0.96S, 1.92S, AND 3.84S, RESPECTIVELY, SINCE ONE ENCODER FRAME IS 60 MS.

	SF w/ NLM	lh-clean	lh-other	RTF
Baseline		3.47	6.41	0.018
N-best resc. (N=10)		3.19	6.03	0.029
Fixed-int. DF (I=64)		3.09	5.78	0.041
Fixed-int. DF (I=32)		3.09	5.73	0.047
Fixed-int. DF (I=16)		3.08	5.73	0.063
Shortest-hyp. DF		3.05	5.68	0.115
Baseline w/ NLM	✓	3.15	5.95	0.082
N-best resc. (N=10)	✓	3.07	5.79	0.089
Fixed-int. DF (I=64)	✓	3.05	5.70	0.099
Fixed-int. DF (I=32)	✓	3.05	5.68	0.107
Fixed-int. DF (I=16)	✓	3.05	5.67	0.122
Shortest-hyp. DF	✓	3.04	5.65	0.174

operating point considering the WER-RTF trade-off. We also evaluate the combination of LLM delayed fusion with NLM shallow fusion, where the LM weight was evenly distributed across the two LMs during decoding, but the final hypothesis was selected with only the E2E and LLM scores (in line 13 of Algorithm 1). The combined fusion effectively reduces the pruning error due to the delay, although it requires a certain overhead for the NLM.

Figure 3 compares delayed fusion and N-best rescoring performance for different beam sizes in CTC prefix beam search. The results indicate that delayed fusion, (e) & (f), achieves lower WERs and RTFs than N-best rescoring, (c) & (d).

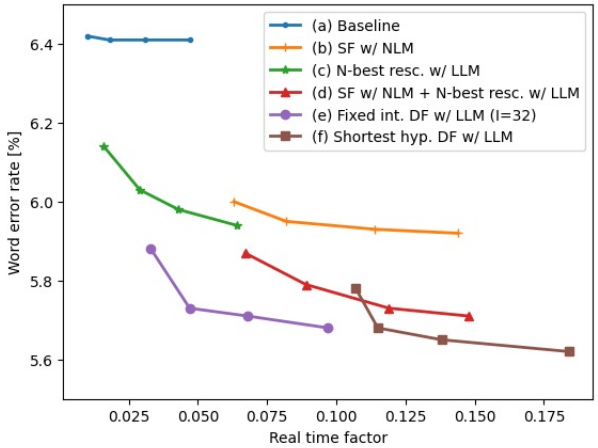

Fig. 3. Delayed fusion vs. N-best rescoring for different beam sizes 5, 10, 15, and 20. WER and RTF were measured on “lh-other”.

Finally, Table III compares shallow and delayed fusion using the same LLM. For shallow LLM fusion, we trained another CTC-AED model from scratch using the LLM tokenizer, which had a 32K vocabulary. In CTC prefix beam search, delayed fusion (shortest hyp.) achieves a good accuracy comparable to or slightly better than shallow fusion, while delayed fusion is 2.2 times faster. In CTC-attention decoding, delayed fusion shows a comparable WER for “lh-clean” but a slightly worse WER for “lh-other”, although it still has a certain speed benefit. However, the advantage of delayed fusion is that it can avoid retraining of ASR models depending on the LLM.

TABLE III

COMPARISON WITH LLM SHALLOW FUSION, WHERE LLM IS OPENLLAMA 3B. FOR SHALLOW LLM FUSION, WE TRAINED ANOTHER CTC-AED MODEL WITH LLAMA TOKENIZER THAT VOCABULARY SIZE WAS 32,000.

Decoding	ASR vocab.	LM fusion	lh-clean	lh-other	RTF
CTC prefix beam search	6K	-	3.47	6.41	0.018
	32K	-	3.61	6.63	0.027
	6K	DF w/ LLM	**3.05**	**5.68**	0.115
	32K	SF w/ LLM	**3.05**	5.75	0.257
CTC-attention decoding	6K	-	2.98	5.64	0.051
	32K	-	3.03	5.62	0.062
	6K	DF w/ LLM	**2.86**	5.35	0.146
	32K	SF w/ LLM	**2.86**	**5.19**	0.169

In summary, LLM delayed fusion achieves 4 - 13% WER reduction (WERR) from the baseline and 3 - 7% WERR from NLM shallow fusion (Table I). Furthermore, it provides lower WERs than N-best rescoring and NLM shallow fusion for the same decoding time (Fig. 3). However, NLM shallow fusion followed by N-best LLM rescoring (plot (d) in Fig. 3) is competitive to delayed fusion ((e) & (f)), where the relative WER difference is around 1 - 3 %. This is a small improvement, but an important advantage of delayed fusion is that it can be used for streaming decoding. In some applications, such as live captioning, lattice/N-best rescoring is not an option, or can have a negative impact on the user experience; delayed fusion is applicable to a wider range of ASR applications.

V. CONCLUSIONS

In this paper, we proposed *delayed fusion*, which applies LLM scores to first-pass ASR hypotheses with a delay during decoding and allows us to easily use pre-trained LLMs in ASR tasks. This method can reduce not only the number of hypotheses scored by the LLM but also the number of LLM inference calls. We can re-tokenize ASR hypotheses during decoding to compute LLM scores if ASR model and LLM employ different tokenizations. We conducted experiments on the LibriHeavy corpus, applying delayed fusion with three public domain LLMs. We demonstrated that (1) Delayed LLM fusion is fast enough compared to standard neural language model (NLM) fusion and (2) Delayed LLM fusion provides lower WERs than N-best LLM rescoring and standard NLM fusion.

Future work will include extensive evaluation of delayed fusion using different datasets, different metrics, e.g., GPU memory consumption, and E2E architectures including RNN Transducers [35].

VI. ACKNOWLEDGEMENTS

We thank David Rybach, Roger Hsiao, Dogan Can, and Pawel Swietojanski for useful suggestions on this work.

REFERENCES

[1] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, “GPT-4 technical report,” *arXiv preprint arXiv:2303.08774*, 2023.

[2] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, “LLaMA: Open and efficient foundation language models,” *arXiv preprint arXiv:2302.13971*, 2023.

[3] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann *et al.*, “PaLM: Scaling language modeling with pathways,” *Journal of Machine Learning Research*, vol. 24, no. 240, pp. 1–113, 2023.

[4] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier *et al.*, "Mistral 7b," *arXiv preprint arXiv:2310.06825*, 2023.

[5] S. Minaee, T. Mikolov, N. Nikzad, M. Chenaghlu, R. Socher, X. Amatriain, and J. Gao, "Large language models: A survey," *arXiv preprint arXiv:2402.06196*, 2024.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. NIPS*, Los Angeles, CA, Dec. 2017, pp. 5998–6008.

[7] P. K. Rubenstein, C. Asawaroengchai, D. D. Nguyen, A. Bapna, Z. Borsos, F. d. C. Quitry, P. Chen, D. E. Badawy, W. Han, E. Kharitonov *et al.*, "AudioPaLM: A large language model that can speak and listen," *arXiv preprint arXiv:2306.12925*, 2023.

[8] 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 *Proc. IEEE ASRU*, 2023, pp. 1–8.

[9] A. Mittal, D. Prabhu, S. Sarawagi, and P. Jyothi, "SALSA: Speedy asr-llm synchronous aggregation," in *Proc. Interspeech*, Kos, Greece, Sep. 2024, pp. 3485–3489.

[10] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "LLaMA 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[11] X. Geng and H. Liu, "OpenLLaMA: An open reproduction of LLaMA," May 2023. [Online]. Available: https://github.com/openlm-research/open_llama

[12] R. Prabhavalkar, T. Hori, T. N. Sainath, R. Schlüter, and S. Watanabe, "End-to-end speech recognition: A survey," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023.

[13] T. Hori, S. Watanabe, Y. Zhang, and W. Chan, "Advances in joint CTC-attention based end-to-end speech recognition with a deep CNN encoder and RNN-LM," in *Proc. Interspeech*, Aug. 2017.

[14] S. Toshniwal, A. Kannan, C.-C. Chiu, Y. Wu, T. N. Sainath, and K. Livescu, "A comparison of techniques for language model integration in encoder-decoder speech recognition," in *Proc. IEEE SLT*, Athens, Greece, Dec. 2018, pp. 369–375.

[15] W. Zhou, Z. Zheng, R. Schlüter, and H. Ney, "On language model integration for RNN transducer based speech recognition," in *Proc. IEEE ICASSP*, Singapore, May 2022, pp. 8407–8411.

[16] A. Sriram, H. Jun, S. Satheesh, and A. Coates, "Cold fusion: Training seq2seq models together with language models," in *Proc. Interspeech*, Hyderabad, India, Sep. 2018, pp. 387–391.

[17] F. Stahlberg, J. Cross, and V. Stoyanov, "Simple fusion: Return of the language model," in *WMT 2018*, Belgium, Brussels, Oct. 2018, pp. 204–211.

[18] 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 *Proc. IEEE SLT*, Shenzhen, China, Dec. 2020, pp. 243–250.

[19] T. Hori, C. Hori, Y. Minami, and A. Nakamura, "Efficient WFST-based one-pass decoding with on-the-fly hypothesis rescoring in extremely large vocabulary continuous speech recognition," *IEEE Transactions on audio, speech, and language processing*, vol. 15, no. 4, pp. 1352–1365, 2007.

[20] H. Sak, M. Saraclar, and T. Güngör, "On-the-fly lattice rescoring for real-time automatic speech recognition," in *Proc. Interspeech*, Sep. 2010, pp. 2450–2453.

[21] A. Y. Hannun, A. L. Maas, D. Jurafsky, and A. Y. Ng, "First-pass large vocabulary continuous speech recognition using bi-directional recurrent DNNs," Dec. 2014, arXiv:1408.2873.

[22] T. Hori, J. Cho, and S. Watanabe, "End-to-end speech recognition with word-based RNN language models," in *Proc. IEEE SLT*, Athens, Greece, Dec. 2018, pp. 389–396.

[23] H. Huang and F. Peng, "An empirical study of efficient ASR rescoring with transformers," *arXiv preprint arXiv:1910.11450*, 2019.

[24] L. Xu, Y. Gu, J. Kolehmainen, H. Khan, A. Gandhe, A. Rastrow, A. Stolcke, and I. Bulyko, "RescoreBERT: Discriminative speech recognition rescoring with BERT," in *Proc. IEEE ICASSP*, Singapore, May 2022, pp. 6117–6121.

[25] W. R. Huang, C. Allauzen, T. Chen, K. Gupta, K. Hu, J. Qin, Y. Zhang, Y. Wang, S.-Y. Chang, and T. N. Sainath, "Multilingual and fully non-autoregressive asr with large language model fusion: A comprehensive study," in *Proc. IEEE ICASSP*. IEEE, 2024, pp. 13 306–13 310.

[26] T. Udagawa, M. Suzuki, G. Kurata, N. Itoh, and G. Saon, "Effect and analysis of large-scale language model rescoring on competitive asr systems," in *Proc. Interspeech*, Incheon, Korea, Sep. 2022, pp. 3919–3923.

[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 *Proc. IEEE ASRU*, Taipei, Dec. 2023, pp. 1–8.

[28] R. Ma, M. J. Gales, K. M. Knill, and M. Qian, "N-best T5: Robust asr error correction using multiple input hypotheses and constrained decoding space," in *Proc. Interspeech*, Aug. 2023.

[29] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*, Pittsburgh, PA, Jun. 2006, pp. 369–376.

[30] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," *Proc. EMNLP*, p. 66, 2018.

[31] W. Kang, X. Yang, Z. Yao, F. Kuang, Y. Yang, L. Guo, L. Lin, and D. Povey, "Libriheavy: a 50,000 hours asr corpus with punctuation casing and context," in *Proc. IEEE ICASSP*, 2024, pp. 10 991–10 995.

[32] S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi, "Hybrid CTC/attention architecture for end-to-end speech recognition," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1240–1253, 2017.

[33] Z. Yao, D. Wu, X. Wang, B. Zhang, F. Yu, C. Yang, Z. Peng, X. Chen, L. Xie, and X. Lei, "WeNet: Production oriented streaming and non-streaming end-to-end speech recognition toolkit," in *Proc. Interspeech*, Brno, Czechia, Sep. 2021, pp. 4054–4058.

[34] T. Hori, S. Watanabe, and J. Hershey, "Joint CTC/attention decoding for end-to-end speech recognition," in *Proc. ACL*, Vancouver, BC, Canada, Jul. 2017, pp. 518–529.

[35] A. Graves, "Sequence transduction with recurrent neural networks," in *Proc. ICML*, Edinburgh, Scotland, Jun. 2012.