MTLM: AN INNOVATIVE LANGUAGE MODEL TRAINING PARADIGM FOR ASR

Qingliang Meng, Pengju Ren, Tian Li, Changsong Dai

Shumei AI Research Institute, Beijing, China

ABSTRACT

Pre-training Transformer-based language models (LMs) on a large amount of text has proven crucial for improving automatic speech recognition (ASR) performance. Generally, traditional LMs are unidirectional and unable to access the context on the right. This paper proposes a method for training LMs that enable traditional unidirectional LMs to fully utilize left and right contexts. Compared with the unidirectional LMs, our LM facilitates ASR to transcribe hypotheses more consistently and in a more semantically unambiguous way, as it incorporates richer contextual representations. Finally, our experimental results on the LibriSpeech corpus demonstrate that our model outperforms traditional unidirectional LMs, whether n-best rescoring or shallow fusion is used as the decoding algorithm.

***Index Terms*—** automatic speech recognition, unidirectional LM, bidirectional contextual representations.

1. INTRODUCTION

End-to-end automatic speech recognition (ASR) is a technology that converts the speech sequence into the target token sequence, which can be trained using paired speech and transcripts [1]. Recently, external language models (LMs) trained on unpaired text data are frequently used to improve the performance of ASR systems [2–6]. Meanwhile, LMs exhibit a wide range of variations depending on pre-training tasks. The most classic pre-training tasks are unidirectional Language Modeling (ULM) and Masked Language Modeling (MLM) [7]. The models trained using these two methods are called unidirectional LMs and bidirectional LMs, respectively. Moreover, the n-best rescoring and shallow fusion are widely used in ASR decoding [8].

In our research, we are surprised that most studies are constrained by the standard decoding method. Thus, the training mode of LMs in ASR is restricted to unidirectional training [8]. In fact, from the perspective of linguistic information capture, both forward and backward training methods can be used to learn language context relevance [2, 9, 10]. For example, english has both active and passive voice. Training with unidirectional LMs for passive voice is unpleasant, as the causal logic of the passive voice needs to be learned from back to front. Regarding the efficiency of enabling LMs to learn contextual semantic relevance, the bidirectional training method should be more effective than unidirectional training [10]. Therefore, we propose a language model (LM) training paradigm for ASR that not only preserves which the LMs can be firmly coordinated with the acoustic model (AM) but also enhances the LMs' training efficiency, completeness in semantic capture, and feasibility in various decoding processes. In this paper, our core contributions are as follows:

i) On the premise of retaining the ULM, we add the MLM training task. This task aims to make the semantic understanding of the LMs benefit not only from unidirectional information but also from bidirectional information. This further advances the understanding of the context semantics of the model, increasing the final word accuracy when combined with AM.

ii) Since the ULM and MLM training tasks are essentially different, it is hard to integrate them into training [4, 9–11]. Therefore, we designed the unidirectional Masked Language Modeling (UMLM) task to reduce this disparity and boost the integration of the knowledge learned from those two tasks. UMLM training uses the same left-to-right logic as ULM training, but the realization scheme of the goal is like MLM. This UMLM task facilitates the cross-flow of information the model learns from different tasks, especially ULM and MLM.

iii) Our model can be used for shallow fusion and rescoring because we still save the output of the unidirectional LM.

On the LibriSpeech dataset [12], compared with the traditional unidirectional LMs for rescoring, our model uses the shallow fusion algorithm to reduce the Word Error Rate (WER) from 3.18% to 2.63% on the *test-clean* dataset and from 8.78% to 7.08% on the *test-other* dataset. Meanwhile, the number of error types on these two datasets has also been reduced by 1/6. These performance improvements demonstrate the superiority of our training paradigm.

2. RELATED WORK

There have been several ways to improve ASR using LMs. [3, 10] propose to use BERT [13] and other bidirectional LMs for n-best rescoring. By predicting pseudo-log-likelihood based on regression, [14] reduces the computing cost of bidirectional LMs. [9] proposes an improvement on bidirectional LMs rescoring, using the ELECTRA model with faster inference than BERT. However, those studies cannot do shallow fusion decoding as they trained LM only using bidirectional logic. [4] leverages knowledge distillation to distill the conditional probabilities of the bidirectional LMs to those of the unidirectional LMs, allowing the model to do shallow fusion. However, since the conditions of bidirectional LMs and unidirectional LMs are incompatible, this operation is unreasonable.

Additionally, [11] train the GPT [15], BERT, and other LMs individually and perform rescoring by merging LMs scores. Nevertheless, this method requires multiple LM models from different training tasks used in decoding, resulting in high computation costs. Our model only requires a single set of parameters to achieve unidirectional and bidirectional LMs and one forward pass computation to obtain the LM score. At the same time, [11] only explicitly adds the scores of each LM to obtain the final model score. This simple fusion method will limit the ability of the model. We do implicit fusion in the training phase and design additional training tasks to compensate for the difference between unidirectional training tasks and MLM. This allows our model to be more flexible and perform better in the ASR system.

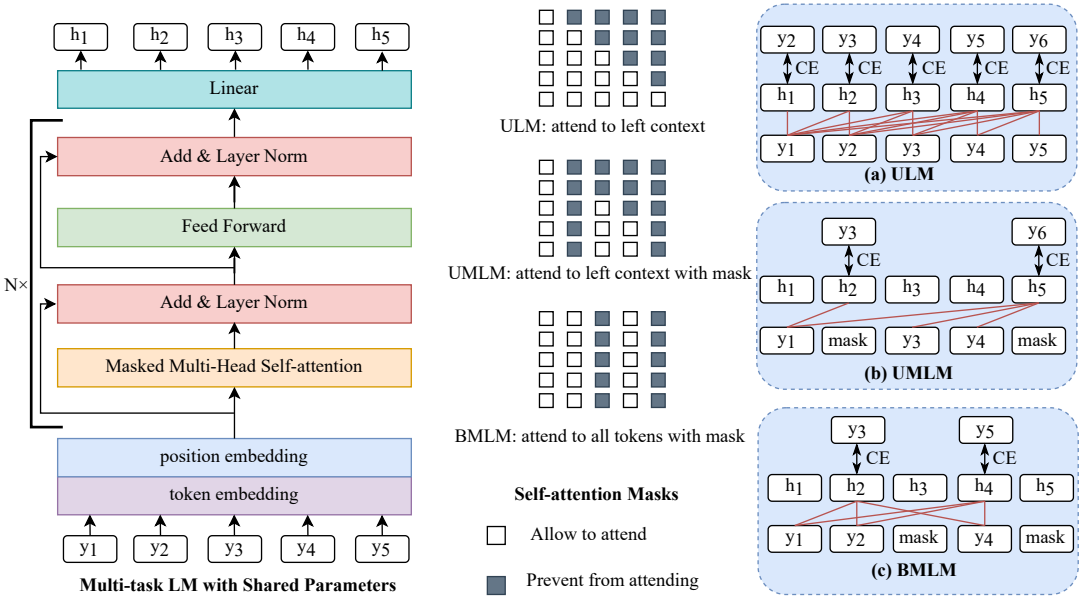

Fig. 1. Illustration of our proposed model. The model structure is based on the Transformer encoder and shares parameters with the three tasks.

3. METHODOLOGY

3.1. Model Architecture

The model architecture is inspired by GPT [15] but uses the encoder of the Transformer [16] rather than the decoder, which is shown in Figure 1. The input $\boldsymbol{y} = y_1, y_2, y_3, ..., y_n$ is a sequence of tokens. We use the one-hot vector I_{y_i} to represent y_i, and $I_{\boldsymbol{y}}$ to represent $\boldsymbol{y}$. In our experiments, y_1 is a special sos token. For input $\boldsymbol{y}$, $\boldsymbol{e} = I_{\boldsymbol{y}} W_e^T + W_p$, where $\boldsymbol{e}$ is the initial input of the encoder module after the matrix calculation of token embedding matrix W_e and position embedding matrix W_p for $\boldsymbol{y}$. $t^l = encoder(t^{l-1}) \quad l \in [1, n]$. We assume that $t^0 = \boldsymbol{e}$ is passed to the model shown in Figure 1. The model is composed of multiple identical layers stacked. Each layer is composed of a multi-head attention layer and a feedforward layer [16]. In addition, the model also uses the residual connection and layer normalization for efficient training [16]. The model finally undergoes the W_e transformation to generate the output distribution $h = t^l W_e$ on target tokens.

3.2. Proposed LM Training Method

We aim to train an LM that can accurately estimate sentence probabilities or predict the next output successfully based on the context and current input. For this reason, we design three training tasks to train the LM by adopting multi-task learning [17, 18]. We refer to this model as MTLM. The details of the training tasks are shown below.

ULM: In ASR, the goal of the LM is to provide a score for hypothesis, which is usually the joint probability $P(\boldsymbol{y})$ assigned by the LM [19]. In this study, the ULM task is a left-to-right language modeling task, which use the complete previous token sequences as context to predict the current token. The optimization goal is shown in Equation (1):

$$\mathcal{L}_{\text{ULM}}(\boldsymbol{y}, \boldsymbol{\theta}) = \mathbb{E}\left(\sum_{i=1}^{n} -\log P\left(y_i \mid y_1, y_2, ..., y_{n-1}\right)\right) \quad (1)$$

where $(y_1, y_2, ..., y_{n-1})$ is the context. The MTLM model trained by this task can predict the probability $P_{LM}(y_t|\boldsymbol{y}_{<t})$ in a single inference pass. Therefore, $P(\boldsymbol{y}) = \sum_{t=1}^{\boldsymbol{n}} \log P_{\text{LM}}\left(y_t \mid \boldsymbol{y_{<t}}\right)$, which requires N inferences from the unidirectional LM. In fact, due to the transformer structure employed by MTLM, the self-attention mechanism allows MTLM to calculate $P_{LM}(y_t|\boldsymbol{y}_{<t})$ for all t in parallel. Therefore, the MTLM model naturally applies to the rescoring and shallow fusion in ASR.

We design the self-attention mask matrix shown in Figure 1 to implement the ULM task. The representation of each token only considers the left context and itself. In the self-attention mask, the shaded part is set to $-\infty$ to prevent attention, and the others are set to 0. The final optimization goal is to minimize the sum of the cross-entropy of $h = \{h_1, ..., h_5\}$ and the corresponding target sequence $y = \{y_2, ..., y_6\}$ where y_6 is eos, as shown in Figure 1.(a).

UMLM: ULM task implements a unidirectional LM, often used as an external LM in the ASR system [8]. Typically, when unidirectional LMs predict the current token, they rely on the knowledge of the preceding token. We argue that if the previous token is incorrectly predicted, it will have an effect on the prediction of the present t-th token. In other word, the probability $P_{LM}(y_t|\boldsymbol{y}_{<t})$ cannot be well predicted. Therefore, we are inspired by the MLM task to randomly mask the preceding partial tokens during model training. The purpose is to allow the model to predict the current token with problematic preceding information. The optimization goal of UMLM is shown in Equation (2):

$$\mathcal{L}_{\text{UMLM}}(\boldsymbol{y}, \boldsymbol{\theta}) = \mathbb{E}\left(\sum_{y_i \in m} -\log p\left(y_i \mid y_1 ... y_{i-1}; \boldsymbol{y}^{\text{m}}\right)\right) \quad (2)$$

where $\boldsymbol{y}^{\text{m}}$ represents replacement operation and m represents a set of the masked token after the input performed y^m. Therefore, in decoding, the MTLM trained on this UMLM task can predict $P_{LM}(y_t|\boldsymbol{y}_{<t})$ more accurately than the unidirectional LM. The self-attention mask is basically the same as the ULM task, except that masked tokens in the previous sequence cannot contribute to the attention computation. The final optimization goal is to minimize the sum of the cross-entropy of $h = \{h_2, h_5\}$ and the corresponding target $y = \{y_3, y_6\}$, as shown in Figure 1.(b).

BMLM: The ULM and UMLM tasks we have implemented up to this point have solely utilized the left context. In ASR, we believe that when generating the probability $P_{LM}(y_t|\boldsymbol{y}_{<t})$, if the bidirectional context information can be combined, it is beneficial for improving the performance [4]. Therefore, we created the Bidirectional

Masked Language Modeling (BMLM) task. The objective function of BMLM is Equation (3):

$$\mathcal{L}_{\mathrm{BMLM}}\left(\boldsymbol{y}, \boldsymbol{\theta}\right)=\mathbb{E}\left(\sum_{y_i \in m}-\log p\left(y_i \mid y_1 \ldots y_n ; \boldsymbol{y}^{\mathrm{m}}\right)\right) \quad (3)$$

As illustrated in Figure 1.(c), the optimization target is the sum of the cross-entropy of h_2 and y_3, h_4 and y_5 in training. In decoding, the MTLM model trained by the BMLM task can incorporate bidirectional contextual information from the left and right sides. In addition, the BMLM task uses a fully open self-attention combined with the MLM task, in which each token can attend to all unmasked tokens.

The overall training task of the MTLM model consists of ULM, UMLM, and BMLM. The parameters of the MTLM are learned to minimize the sum of the cross-entropy losses of these training tasks and shared among all tasks. The final loss function is shown in Equation (4).

$$\min_{\theta}(\mathcal{L}_{\mathrm{ULM}}(\boldsymbol{y}, \theta)+\mathcal{L}_{\mathrm{UMLM}}(\boldsymbol{y}, \theta)+\mathcal{L}_{\mathrm{BMLM}}(\boldsymbol{y}, \theta)) \quad (4)$$

3.3. Decoding

There are two ways to integrate external LMs into an end-to-end ASR model: rescoring and shallow fusion [9, 20, 21]. In rescoring, the ASR model generates n-best list through the beam search, and then the LMs rerank each hypothesis in the list. Score $_{\mathrm{LM}}(\boldsymbol{y})$ is the score of LM for hypothesis $\boldsymbol{y}$, which is often expressed by the $log\text{-}likelihood$, as shown in Equation (5):

$$\mathrm{Score}_{\mathrm{LM}}(\boldsymbol{y})=\sum_{t=1}^{\boldsymbol{n}} \log P_{\mathrm{LM}}\left(y_t \mid \boldsymbol{y}_{<\boldsymbol{t}} ; \boldsymbol{\theta}\right) \quad (5)$$

where $\boldsymbol{y}_{<t}=\left(y_1, ..., y_{t-1}\right)$ and $P_{LM}(y_t|\boldsymbol{y}_{<t})$ is the conditional probability of predicting the current token given the prior tokens. For unidirectional LMs, $P_{LM}(y_t|\boldsymbol{y}_{<t})$ can be calculated by multiplying the output probabilities of each token in the sequence. For bidirectional LMs, it can predict current token y_t based on the left and right context [6]. The output of bidirectional LMs can be used to estimate the conditional probability $P_{\mathrm{LM}}\left(\boldsymbol{y}_t \mid \boldsymbol{y}_{\backslash t}\right)$, where $\boldsymbol{y}_{\backslash t}=\left(y_1, ..., y_{t-1}, [\mathrm{mask}], y_{t+1}, ..., y_{\boldsymbol{n}}\right)$ [6]. $\boldsymbol{y}_{\backslash t}$ represents the sentence in which the t-th token is replaced by a special $[\mathrm{mask}]$ token. Therefore, Equation (6) is usually used directly to represent the score of bidirectional LMs for hypothesis $\boldsymbol{y}$ [6].

$$\mathrm{Score}_{\mathrm{LM}}(\boldsymbol{y})=\sum_{t=1}^{\boldsymbol{n}} \log P_{\mathrm{LM}}\left(y_t \mid \boldsymbol{y}_{\backslash t} ; \boldsymbol{\theta}\right) \quad (6)$$

In shallow fusion, [7, 11, 12] generally uses LM to perform log-linear interpolation in each step of beam search.

$$\begin{aligned} y_t=& \arg\max_{y_t} \log P_{AM}(y_t \mid X, \boldsymbol{y}_{<t}) \\ &+\lambda \log P_{LM}(y_t \mid \boldsymbol{y}_{<t}) \end{aligned} \quad (7)$$

The shallow fusion can better integrate the LM into the ASR system, because the generation of each token requires the participation of LM and AM [8].

After training the LM separately, we integrate our LM into the beam search algorithm. For decoding, we follow the joint CTC/S2S one-pass decoding algorithm [22]. However, we have two differences. The first is that [22] uses a fixed maximum length to determine the hypothesis length of audio transcription. We employ a prediction method based on CTC greedy search to find the optimum transcript length for each audio sample. This conserves a substantial amount of computational resources during decoding. Second, we prune finished and unfinished hypothesis routes together, whereas [27] never prunes finished routes. Without pruning, short sentences are more likely to remain in the finished pathways collection since they have higher scores, restricting final performance. According to the design of our training task, our MTLM model is suitable for both n-best rescoring and shallow fusion.

4. EXPERIMENTS

4.1. Experimental Setup

We evaluate our method using 1000 hours of audio data and transcripts from the LibriSpeech corpus [12]. The training set is divided into 100, 360, and 500 hours, while the development and test sets are divided into clean and other categories. The training data are used to train the AM. This part of the speech data corresponds to the transcribed text data and the additional text data of 800 million words provided by LibriSpeech are used as the training corpus of the LM.

In our experiments, we implement the AM following the research of [23], which is a CTC+S2S hybrid architecture. CTC and S2S share the same encoder, and S2S has a separate decoder. The encoder consists of 12 Transformer blocks [16]. Each block has an 8-head self-attention layer and a 2048-dimensional feed-forward sub-layer. The decoder is a 6-layer transformer decoder block. We employ a 100-channel filter bank with a 3-dimension pitch for the input. We use byte-pair encoding (BPE) [24] to build 7002-sized subwords set, including sos, eos, and blank labels. Table 1 shows the WER obtained from our AM. The AM selects the hypothesis with the highest score from the candidate hypotheses. The WER is 3.4% on *test-clean* and 9.04% on *test-other*. Our MTLM model has six 12-head Transformer blocks. The self-attention dimension is 768, and the feed-forward network dimension is 3072 in each Transformer block. The same 7002-sized word vocabulary with the AM is used in LM. We train the LM from scratch used the text-only data of LibriSpeech corpus and optimized by Adam optimizer [25], where β_1=0.9, β_2=0.999, warmup steps are 5000, the maximum learning rate is $2e^{-4}$ and decay to $1e^{-6}$.

4.2. Results and Discussion

In this section, we compare the WER performance of the ASR system after integrating the AM with the traditional unidirectional LM (UNILM) or our MTLM model by n-best rescoring or shallow fusion. Table 1 summarizes the WER result. On the LibriSpeech dataset [12], we train the UNILM and MTLM models from scratch. The UNILM model is trained using the ULM task specified in Section 3.2, frequently utilized in ASR.

The first row of Table 1 represents the WER performance of AM. The following two parts are the WER of the ASR system using different decoding strategies after the introduction of the UNILM and the MTLM model, respectively. The experimental results in Table 1 demonstrate that regardless of the decoding strategy used, the UNILM or MTLM models outperform the AM without the addition of LM. Particularly, the MTLM model can generate the likelihood score of Equation (5) based on ULM task, and it can also generate the pseudo log-likelihood score of Equation (6) based on BMLM task. Therefore, the MTLM can do n-best rescoring in these two ways. The MTLM model has a WER of 2.63% on *test-clean* and a WER of 7.08% on *test-other*, both exceeding UNILM.

As a result, Table 1 demonstrates the efficacy of introducing MBLM techniques. Simultaneously, we believe that the distinction

between the two decoding methods is that the LMs can participate in scoring after the AM has generated an entire path of beam size in n-best rescoring [8, 26]. Therefore, the AM alone determines the generation of hypotheses, and the rescoring method is constrained. When each character is generated with shallow fusion, the LM contributes to the scoring in the decoding [21, 27]. The final beam size hypotheses are assumed to be generated jointly by the AM and LMs, so the WER performance is better.

Table 1. The WER (%) performance of UNILM and MTLM models using various decoding methods (lower is better). † means that the model uses Equation (5) for n-best rescoring, ♦ means that the model uses Equation (6) for n-best rescoring, and ♣ means that the model uses the shallow fusion decoding. (Beam Size=3)

Model	dev		test	
	clean	other	clean	other
AM	3.18	8.94	3.4	9.04
+UNILM†	3.02	8.69	3.18	8.78
+UNILM♣	2.57	6.99	2.81	7.36
+MTLM†	3.01	8.67	3.14	8.75
+MTLM♦	3.03	8.67	3.14	8.76
+MTLM♣	**2.46**	**6.75**	**2.63**	**7.08**

4.3. Error Type Analysis

In this section, we merge the *test-clean* and *test-other* data into one dataset. Then according to the number of words in the transcript corresponding to each audio, this dataset is divided into 3 categories. As shown in Table 2, long, medium, and short correspond to a word count larger than 20, between 10 and 20, and less than 10 words, respectively. The AM in Table 2 represents the baseline, and the subsequent four numbers reflect the number of deletions, insertions, replacements, and overall errors of the AM.

Table 2. The number of error types for AM, UNILM and MTLM (lower is better). The experimental results are classified into three categories: long (L), medium (M), and short (S), and the error types are deletion, insertion, replacement, and overall error. (Beam Size=3)

Len	Model	Del	Ins	Sub	Overall
L	AM	374	284	2650	3308
	+UNILM†	320	274	2541	3135
	+MTLM†	**319**	273	2531	3123
	+MTLM♦	320	272	2533	3125
	+UNILM♣	370	205	2079	2654
	+MTLM♣	367	**199**	**1932**	**2498**
M	AM	207	165	1620	1992
	+UNILM†	204	166	1565	1935
	+MTLM†	203	164	1558	1925
	+MTLM♦	204	166	1559	1929
	+UNILM♣	203	116	1297	1616
	+MTLM♣	**190**	**110**	**1260**	**1560**
S	AM	150	104	975	1229
	+UNILM†	154	105	943	1202
	+MTLM†	153	103	936	1192
	+MTLM♦	149	102	942	1193
	+UNILM♣	**135**	89	855	1079
	+MTLM♣	143	**78**	**826**	**1047**

Table 2 demonstrates that only in the short category when shallow fusion is used as the decoding algorithm, the MTLM is inferior to the UNILM model. The data shows that the number of error types is 143 for MTLM, while UNILM is 135. Nevertheless, in all other circumstances, MTLM can further reduce the number of errors of the UNILM. In particular, the MTLM significantly and consistently outperforms UNILM in reducing the number of overall errors.

4.4. The GuideScore Algorithm

Table 3. The difference between whether LM participates in the GuideScore algorithm. (Beam Size=3)

GuideScore	Model	dev		test	
		clean	other	clean	other
S2S	UNILM♣	2.57	6.99	2.81	7.36
	MTLM♣	**2.46**	**6.75**	**2.63**	**7.08**
LM+S2S	UNILM♣	2.58	6.98	2.82	7.3
	MTLM♣	**2.46**	6.77	2.64	7.1

In decoding, there is a $GuideScore$ algorithm in [22]. This algorithm is responsible for generating the next set of candidate characters for the hypothesis. This algorithm can use the joint score of S2S and LM, or only use the score of S2S. All experimental data in Table 1 and 2 are obtained solely by using the S2S model score from the AM as the hypothesis score basis for decoding. Therefore, we conducted an experiment to study the performance changes of the $GuideScore$ algorithm after introducing LM participation. As shown in Table 3, when both the LM and the S2S model are used in the $GuideScore$, the WER performance of the UNILM and MTLM models on both the *test-clean* and *test-other* datasets is slightly lower than that of the S2S model alone. The reason may be that the excessive participation of the LM interferes with the judgment of the AM and makes the entire ASR system pay more attention to the LM.

4.5. The Effect of Different Beam Sizes

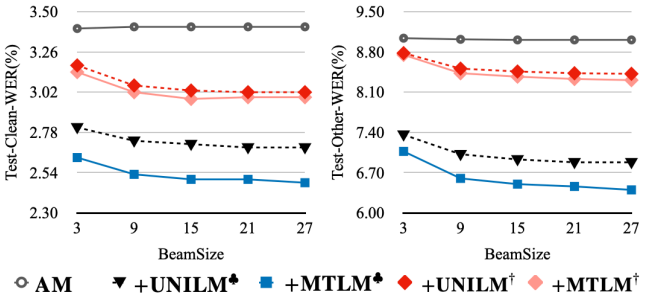

Fig. 2. The WER performance of the AM, UNILM and MTLM on the *test-clean* and *test-other* datasets with different beam sizes.

Finally, as illustrated in Figure 2, we compare the WER of the AM, the UNILM, and the MTLM model when beam size varies. No matter which decoding method is used, MTLM outperforms UNILM and AM. Additionally, the WER decreases regardless of decoding method as beam size increases. Among them, shallow fusion achieves the most significant performance improvement.

5. CONCLUSIONS

In this paper, we propose an LM training method that allows the model to be used for rescoring and shallow fusion, and the model can also learn bidirectional information. We also make a detailed comparison of model performance. Based on experiments on LibriSpeech, we demonstrate that the MTLM model can further improve the ASR accuracy compared to UNILM. Finally, we also conduct experiments on error type to evaluate the specific advantages of the MTLM. In future work, we will investigate alternative methods for training LMs.

6. REFERENCES

[1] Anchal Katyal, Amanpreet Kaur, and Jasmeen Gill, "Automatic speech recognition: a review," *International Journal of Engineering and Advanced Technology (IJEAT)*, vol. 3, no. 3, pp. 71–74, 2014.

[2] Dominique Fohr and Irina Illina, "Bert-based semantic model for rescoring n-best speech recognition list," in *INTERSPEECH 2021*, 2021.

[3] Pablo Ortiz and Simen Burud, "Disambiguation-bert for n-best rescoring in low-resource conversational asr," *arXiv preprint arXiv:2110.02267*, 2021.

[4] Hayato Futami, Hirofumi Inaguma, Sei Ueno, Masato Mimura, Shinsuke Sakai, and Tatsuya Kawahara, "Distilling the knowledge of bert for sequence-to-sequence asr," *arXiv preprint arXiv:2008.03822*, 2020.

[5] Xi Chen, Songyang Zhang, Dandan Song, Peng Ouyang, and Shouyi Yin, "Transformer with bidirectional decoder for speech recognition," *arXiv preprint arXiv:2008.04481*, 2020.

[6] Takuma Udagawa, Masayuki Suzuki, Gakuto Kurata, Nobuyasu Itoh, and George Saon, "Effect and analysis of large-scale language model rescoring on competitive asr systems," *arXiv preprint arXiv:2204.00212*, 2022.

[7] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang, "Pre-trained models for natural language processing: A survey," *Science China Technological Sciences*, pp. 1–26, 2020.

[8] Anjuli Kannan, Yonghui Wu, Patrick Nguyen, Tara N Sainath, Zhijeng Chen, and Rohit Prabhavalkar, "An analysis of incorporating an external language model into a sequence-to-sequence model," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 1–5828.

[9] Hayato Futami, Hirofumi Inaguma, Masato Mimura, Shinsuke Sakai, and Tatsuya Kawahara, "Asr rescoring and confidence estimation with electra," *arXiv preprint arXiv:2110.01857*, 2021.

[10] Joonbo Shin, Yoonhyung Lee, and Kyomin Jung, "Effective sentence scoring method using bert for speech recognition," in *Asian Conference on Machine Learning*. PMLR, 2019, pp. 1081–1093.

[11] Xianrui Zheng, Chao Zhang, and Philip C Woodland, "Adapting gpt, gpt-2 and bert language models for speech recognition," *arXiv preprint arXiv:2108.07789*, 2021.

[12] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[14] Julian Salazar, Davis Liang, Toan Q Nguyen, and Katrin Kirchhoff, "Masked language model scoring," *arXiv preprint arXiv:1910.14659*, 2019.

[15] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever, "Improving language understanding by generative pre-training," 2018.

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[17] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon, "Unified language model pre-training for natural language understanding and generation," *arXiv preprint arXiv:1905.03197*, 2019.

[18] Fang Liu, Ge Li, Yunfei Zhao, and Zhi Jin, "Multi-task learning based pre-trained language model for code completion," in *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, 2020, pp. 473–485.

[19] Abhilash Jain, Aku Rouhe, Stig-Arne Grönroos, Mikko Kurimo, et al., "Finnish asr with deep transformer models.," in *Interspeech*, 2020, pp. 3630–3634.

[20] Kazuki Irie, Albert Zeyer, Ralf Schlüter, and Hermann Ney, "Language modeling with deep transformers," *arXiv preprint arXiv:1905.04226*, 2019.

[21] Jan Chorowski and Navdeep Jaitly, "Towards better decoding and language model integration in sequence to sequence models," *arXiv preprint arXiv:1612.02695*, 2016.

[22] Shinji Watanabe, Takaaki Hori, Suyoun Kim, John R Hershey, and Tomoki 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.

[23] Tian Li, Qingliang Meng, and Yujian Sun, "Improved noisy iterative pseudo-labeling for semi-supervised speech recognition," in *2023 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2023.

[24] Taku Kudo, "Subword regularization: Improving neural network translation models with multiple subword candidates," *arXiv preprint arXiv:1804.10959*, 2018.

[25] Ilya Loshchilov and Frank Hutter, "Fixing weight decay regularization in adam," 2018.

[26] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio, "End-to-end attention-based large vocabulary speech recognition," in *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2016, pp. 4945–4949.

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