Interspeech 2022
18-22 September 2022, Incheon, Korea

DOMAIN PROMPTS: TOWARDS MEMORY AND COMPUTE EFFICIENT DOMAIN ADAPTATION OF ASR SYSTEMS

Saket Dingliwal, Ashish Shenoy, Sravan Bodapati, Ankur Gandhe, Ravi Teja Gadde, Katrin Kirchhoff*

{skdin, ashenoy, sravanb, aggandhe, gadderav, katrinki}@amazon.com

ABSTRACT

Automatic Speech Recognition (ASR) systems have found their use in numerous industrial applications in very diverse domains creating a need to adapt to new domains with small memory and deployment overhead. In this work, we introduce *domain-prompts*, a methodology that involves training a small number of domain embedding parameters to prime a Transformer-based Language Model (LM) to a particular domain. Using this domain-adapted LM for rescoring ASR hypotheses can achieve 7-13% WER reduction for a new domain with just 1000 unlabeled textual domain-specific sentences. This improvement is comparable or even better than fully fine-tuned models even though just 0.02% of the parameters of the base LM are updated. Additionally, our method is deployment-friendly as the learnt domain embeddings are prefixed to the input to the model rather than changing the base model architecture. Therefore, our method is an ideal choice for on-the-fly adaptation of LMs used in ASR systems to progressively scale it to new domains.

Index Terms— domain-adaptation, prompt-tuning, gpt2, multi-domain ASR, parameter-efficiency, low-data setting

1. INTRODUCTION

Automatic Speech Recognition (ASR) systems form a key component of various products across industry. With recent advancements [1, 2, 3], they have been deployed in a wide range of domains, including healthcare, travel reservations, and customer services etc. A typical technique to improve the performance of these systems is to do a rescoring of the n-best hypotheses with an external Language Model (LM) [2]. Recent pretrained Transformer-based LMs such as GPT-2 [4] and BERT [5] have been shown [6] to be more effective than conventional LSTM based LMs for rescoring. However, their use in an industrial ASR system that needs to incrementally support new domains poses the following challenge. As showcased in [7, 8], domain-specific data is useful for improving performance in a domain. However, retraining or maintaining copies of Transformer-based LMs for each domain separately is not scalable as updating and storing millions of parameters comes with a large cost. Therefore, a need for an efficient domain-adaptation method for such LMs is evident. [9, 10, 11] used external knowledge, memory and context respectively to improve performance in specific difficult domains, while [12, 13] adapted the neural LM used within the system. However, to the best of our knowledge, ours is the first work to propose and study methods to do efficient domain-adaptation of Transformer-based LMs to benefit ASR systems. Language modeling literature [14, 15, 16] introduced novel methodologies to solve a related problem of efficiently adapting such LMs to specific tasks. Instead of fine-tuning and storing millions of parameters for each task, they propose augmenting the frozen task-agnostic model with a handful of task-specific trainable parameters. For example, AdapterHub [14] introduced new task-specific layers in conjunction to frozen pre-trained weights of LMs. More recent models, such as GPT-3 [17], are able to solve new tasks with the help of just the textual descriptions of the task (called prompts).

Extending [18], the focus of this work is to adapt such LMs to different domains of the same task rather than solving multiple tasks. Our objective is to learn a small set of domain-specific parameters to score ASR hypotheses better than the base Transformer-based LM without the domain data. Drawing ideas from prompt-tuning [15] for task adaptation, we introduce *domain-prompts* for our goal. We define *domain-prompts* as domain-specific embeddings, which when prefixed to the sequence of token embeddings, and passed through a pretrained Transformer LM, return the probability distribution of the next token, close to that given by a fully domain-adapted LM. Our main contributions are summarized as follows: (1) we introduce a new methodology *domain-prompts*, which is the first attempt to apply prompt-tuning for parameter-efficient domain-adaptation of Transformer-based LMs for their use in ASR systems, (2) In new domains with limited data, we demonstrate that rescoring ASR hypotheses with LM adapted using our method can achieve 7-13% WER reduction while using a handful of additional domain-specific parameters (3) Along with saving memory and training cost, *domain-prompts* can match or even beat the performance of fully fine-tuned models with no change to the deployment of the base model, thereby making it the ideal choice for on-the-fly domain adaptation for industrial ASR systems.

*work carried out while working at Amazon

10.21437/Interspeech.2022-824

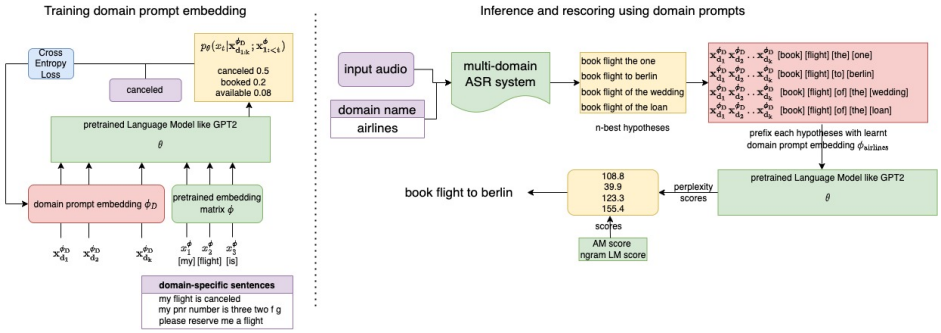

Fig. 1. Domain Prompts: training (left) and inference (right) methodology for domain-adaptation

2. METHODOLOGY

GPT-3 [17] introduced natural-language *prompts* as textual descriptions of a task. For example - prefixing "translate the sentence to French " to the input for the machine translation task. In prompt-tuning [15], rather than designing these prompts manually, the model learns their embeddings using a few labeled examples from the task. We demonstrate that such embeddings can also be learnt for different domains, i.e., we can prefix a sentence with additional domain-specific embedding vectors such that it improves the perplexity of the sentences from that domain. We use the self-supervised task of predicting the next token in unlabeled domain-specific text for training. An unlabelled sentence is a sequence of T tokens $\{x_1, x_2 \dots x_T\}$. Let $\mathbf{x}^{\phi}_{\mathbf{1:T}}$ be the corresponding concatenation of d-dimensional embedding vectors for these tokens, given by the embedding matrix parameterized by ϕ. These vectors are propagated through multiple Transformer layers before taking a softmax to obtain the probability distribution over the vocabulary for the next possible token. Mathematically, we denote the probability of predicting x_t token at t^{th} time-step as $p_\theta(x_t|\mathbf{x}^{\phi}_{\mathbf{1:<t}})$ where θ represents all the parameters in the Transformer layers. Both θ and ϕ have large dimensions and are trained together on a large corpus of text.

In our method, for any domain D, we begin with pretrained $\{\theta, \phi\}$ and introduce a small number of additional parameters ϕ_D in the form of k d-dimensional embedding vectors $[\mathbf{x}^{\phi_\mathbf{D}}_{\mathbf{d_1}}, \mathbf{x}^{\phi_\mathbf{D}}_{\mathbf{d_2}} \dots \mathbf{x}^{\phi_\mathbf{D}}_{\mathbf{d_k}}]$ concatenated together as $\mathbf{x}^{\phi_\mathbf{D}}_{\mathbf{d_{1:k}}}$. We prefix them to each sentence while predicting the next token at each time step. While training, we keep $\{\theta, \phi\}$ fixed and learn ϕ_D by minimizing the cross-entropy loss between the true token value x_t and its predicted probability. Eq. 1 represents the loss for one such sequence and we add the loss value for all the sentences from the domain.

$$\phi_D = \underset{\phi_D}{\operatorname{argmin}} \sum_{t=1}^{t=T} -\log p_\theta(x_t|\mathbf{x}^{\phi_\mathbf{D}}_{\mathbf{d_{1:k}}}; \mathbf{x}^{\phi}_{\mathbf{1:<t}}) \qquad (1)$$

We hypothesize that the self-attention layer in Transformer-based LMs will create an interaction between ϕ_D and the embeddings of the tokens from sentences, thereby improving the score to cater to the domain D. During inference, we prefix the trained *domain-prompts* ($\mathbf{x}^{\phi_\mathbf{D}}_{\mathbf{d_{1:k}}}$) of the corresponding domain to the hypotheses from the ASR system and use perplexity scores from the model for rescoring as explained in Fig. 1. Further, in our implementation, we used gradient descent to minimize the loss and instead of initializing the parameters ϕ_D randomly, we begin with token embeddings (using ϕ from pretrained model) of the k most frequent words in the training sentences of the domain, following prior work [15]. Also, to ensure inference latency does not increase due to additional computations, we use caching to store the state of the Transformer after propagating the k *domain-prompts* through the Transformer layers. These domain-embeddings are constant for all the hypotheses in the domain and hence its forward pass through Transformer layers can be precomputed to ensure the latency for scoring a hypothesis is same for both base and adapted versions of the Transformer LM.

3. EXPERIMENTAL SETUP

To test the effectiveness of the proposed *domain-prompts*, we run extensive experimentation with numerous adaptation baselines with four different domains, model sizes, initializations and training set sizes. We use two versions of GPT-2 architecture [4] as our base models: (1) *gpt2* (117M (million) parameters) (2) *gpt2-medium* (345M parameters). We conduct all our experiments on an AWS ec2 instance with 8 Tesla V100 GPUs using a hybrid ASR system. Such a system consists of an Acoustic Model (AM), and two different LMs. *First pass LM* is an ngram LM which is directly composed with lattice from the AM while *second pass LM* is a Neural LM and is used for rescoring the n-best hypotheses decoded from the lattice. Our AM is trained on 12k hours of audio. Our first pass LM is a 4-gram model with Kneser-Ney (KN) [19] smoothing with vocabulary size of 500k words trained on a heterogeneous corpus of text. We use scores from different second pass LMs (and interpolate with scores from the AM and the first pass LM) to rescore n-best lists with $n = 10$. Our performance metric is Word Error Rate (WER) and we report WER Reduction % (WERR) over the baseline.

Table 1. Generating text from *gpt2* adapted to airlines domain

Input tokens	"hello how are you"
no-adaptation	hello how are you doing? I'm really happy with the results.
full-fine-tuning	hello how are you able to get a new flight I'm flying from London Heathrow to Dubai
domain-prompts	hello how are you able to get a refund on the flight I'm flying from Glasgow to Madrid today

We evaluate under two different settings representative of any industrial multi-domain ASR system: (1) *New domains with limited data*: To simulate the scenario of on-the-fly adaptation to new/unseen domains, we use only 1k domain-specific sentences to adapt the second pass LM. (2) *Domains with large data*: 50k domain-specific sentences are added to the training set of both the first and the second pass LM.

Dataset: For our experiments, we need domain-wise (i) textual data for LM adaptation, (ii) audio data for evaluation. We were not able to find any public dataset that can be split domain-wise and meets both the criteria. Therefore, we use in-house datasets from four different domains with all Personal Identifiable Information (PII) removed. For textual data, we use 1k and 50k domain-specific conversational sentences in the two settings respectively. This data is split into 80:20 as train and dev set respectively. The perplexity on the dev set is used for tuning the hyperparameters like learning rate etc for all the baselines. For evaluation, we use 500 labelled 8khz audios per domain which are single utterances from a conversational task-oriented dialog system.

Baselines: The different choices of second pass LM used for comparison in Table 2 are defined below: (1) *no rescoring* (baseline): Use the 1-best hypothesis with no second pass LM. (2) *LSTM LM*: 2-layer LSTM based LM with embedding dimension (d), hidden dimension (h) and a word-based tokenizer with vocabulary size (V). (3) *no adaptation*: Out-of-the-box Transformer-based LMs without the use of any domain data or any parameter update. (4) *tuning embedding layer*: Update the parameters of the embedding matrix (ϕ) using domain-specific data while keeping θ fixed. (5) *prompt-designing*: Prefix manually-defined prompts to the hypotheses without any training. 20 most frequent words from the domain were used as fixed prompts. (6) *domain-embedding* [7]: Learn the embedding of a special domain token. It is equivalent to $k = 1$ in our method. (7) *domain Adapters*: Adapters [14] are typically used for parameter-efficient task adaptation of Transformer-based LM. However, we train *domain Adapters* here using the self-supervised task of next token prediction with different reduction factors (c). (8) *domain-prompts*: Update the domain embeddings (ϕ_D) for different values of k and initializations of ϕ_D. (9) *full fine-tuning*: Update all the parameters (θ, ϕ) of the base LM. (10) *oracle*: pick hypothesis in the n-best list with minimum WER to know the upper bound of improvements through rescoring.

4. RESULTS AND DISCUSSION

Domain adaptation methods are used to prime the LM to a particular domain. As shown in Table 1, the LM adapted using *domain-prompts* learnt from airlines data, completes sentence to a very domain-specific utterance in contrast to the out-of-the-box LM extending the input to a generic sentence. We showcase some qualitative results in Table 3, where perplexity scores from vanilla *gpt2* and *domain-prompts* ($k = 50$) for similar sounding hypotheses is provided. These examples indicate how domain information helps to disambiguate to choose the right hypothesis. We summarize the WERR scores of all our methods in the two different settings in Table 2. The first column contains the name of the methodology and its hyper-parameter, second column is the base model used while the third column represents the number of domain-specific trainable parameters needed in addition to the base model. Note that the goal of the experiments is not only to find the best performing method, but also to discover settings that achieve optimal performance with the minimal number of additional parameters. This is a critical decision point for systems to scale to potentially hundreds of domains as storage and training cost are directly linked to the number of domain-specific parameters.

The adaptation to domain-specific data is useful for performance in all the domains in both the settings (row 4 vs. 19 or row 3 vs. 18). Even rescoring with *dialog-gpt2-medium* [20] (row 5), which is pretrained on a large dialog corpus is not as effective as adaptation to small amounts of domain data. The WERR numbers vary across different domains but the relative performance for different domain-adaptation methods is consistent across all domains. The domains are fairly different from each other. Domains like healthcare have a large number of unseen technical words and hence improvements through rescoring are relatively small. In the large-data-setting, the domain-specific data is also added to first pass LM and hence the quality of the n-best hypotheses is better, which leads to larger performance improvements through rescoring (row 20). Similar to results in [6], we observe Transformer-based LMs perform better than LSTM based methods (row 1 and 2 vs. 18). For fair comparison, we increase the size of the LSTM model and pretrain it with wiki-text-103 [21] (row 2), but it still cannot match Transformer models. Also, as we increase the size of the Transformer, the performance improves (row 18 vs. 19), further indicating the need for parameter-efficient adaptation methods. Our main conclusions about our method are as follows:

Domain prompts are the most parameter efficient: *domain-prompts* uses $< 0.02\%$ of parameters of the base model per domain to achieve performance comparable to domain-specific fully fine-tuned models with millions of parameters (row 17 vs. 19). Although fine-tuned models perform better than our method in the large-data-setting, their improvement comes at the cost of deploying separate models for each domain. This is expected when adequate data is available, large

Table 2. Comparison of different domain-adaptation methods for different domains in parameter count and WERR% metric.

			# additional	Low data setting (1k sentences)				Large data setting (50k sentences)			
	domain adaptation methods	**base model**	**domain-specific**	**WER Relative % ↑**				**WER Relative % ↑**			
			params	**airlines**	**fastfood**	**healthcare**	**insurance**	**airlines**	**fastfood**	**healthcare**	**insurance**
	no rescoring	-	0	-	-	-	-	-	-	-	-
1	LSTM LM ($d = h = 256, V = 15k$)	LSTM	8.7M	0	0	0	0	4.8	0.8	0	6.2
2	LSTM LM ($d = h = 512, V = 229k$)	wiki103-LSTM	121M	3.1	0	0	3.9	7.3	1.7	0	6.2
3	no adaptation	gpt2	0	0.5	2.0	2.4	2.7	2.4	0.8	0.8	3.3
4	no adaptation	gpt2-medium	0	3.6	3.4	3.4	4.8	4.9	3.4	0.8	8.2
5	no adaptation	dialog-gpt2-medium	0	3.1	1.5	1.0	3.2	4.9	2.6	0	5.1
6	tuning embedding layer	gpt2	40M	3.6	4.9	3.9	6.0	7.3	9.4	0	6.2
7	prompt-designing	gpt2	0	0.5	0	2.4	2.7	3.7	1.7	0	4.1
8	domain-embedding	gpt2	768	0.5	2.0	3.4	4.3	3.7	1.7	0	3.1
9	domain Adapter ($c = 512$)	gpt2	0.3M	4.6	5.9	3.4	5.9	6.1	7.7	0.8	6.2
10	domain Adapter ($c = 16$)	gpt2	1.1M	4.6	7.8	5.8	6.4	8.5	11.1	0.8	8.2
11	domain Adapter ($c = 512$)	gpt2-medium	1M	7.1	10.7	6.3	8.1	9.8	8.6	1.6	10.3
12	domain Adapter ($c = 16$)	gpt2-medium	3.6M	7.6	9.8	4.8	7.0	8.5	12.0	3.2	10.3
13	domain-prompts ($k = 10$, vocab init)	gpt2	7680	3.8	7.3	5.3	3.2	6.1	7.7	3.2	7.2
14	domain-prompts ($k = 50$, random init)	gpt2	0.04M	6.1	8.8	6.3	6.5	6.1	8.6	2.4	8.2
15	domain-prompts ($k = 50$, vocab init)	gpt2	0.04M	5.1	8.8	6.8	7.0	8.5	9.4	4.0	9.2
16	domain-prompts ($k = 200$, vocab init)	gpt2	0.16M	6.1	9.3	7.3	5.9	8.5	9.2	4.0	10.3
17	domain-prompts ($k = 50$, vocab init)	gpt2-medium	0.05M	**8.1**	**13.1**	**7.7**	**8.1**	**11.0**	11.1	5.7	**12.4**
18	full-fine-tuning	gpt2	117M	6.6	9.8	6.8	6.4	8.5	12.9	4.8	12.3
19	full-fine-tuning	gpt2-medium	345M	7.1	11.2	7.2	7.0	8.5	**16.2**	**7.2**	**12.4**
20	oracle	-	-	22.0	32.8	29.8	21.1	39.3	38.6	36.2	34.7

number of domain-specific parameters can capture larger amount of domain information. Adapters which are commonly used for their efficiency, have limited efficacy when compared to our method. *Domain prompts* with 20 times less parameters, can beat its performance (row 11 vs. 17). Further, Adapters have another limitation that their number of parameters scale with the number of layers in the base Transformer model (*gpt2* vs.*gpt2-medium*) while *domain-prompts* depends only on the embedding dimension d. Fine-tuning a subset of parameters in Transformer-based LM (row 6) is not effective as manually selecting a subset of most influential parameters is difficult and performs worse than our method in both performance and cost. Methods like fixing prompts or training a single domain embedding vector use no or very small number of parameters but their improvements are only marginal over unadapted base LM (row 3 vs. 7 and 8).

Domain prompts achieves best performance for new/unseen domains: This setting represents common practical applications where a new domain with limited amount of available data needs to be added to the ASR system. Here, *domain-prompts* can reap both the benefits: (1) rich pretraining of Transformer based LM (2) no overfitting on limited number of examples. This is evident from fact that fine-tuned *gpt2* performs slightly better than corresponding prompt-tuned version (row 16 vs. 18) while opposite is true for *gpt2-medium* (row 17 vs. 19) indicating updating large number of parameters is prone to overfitting. Hence, *domain-prompts* presents an ideal case to capture all the necessary domain specific information from 1k examples in its limited domain-specific parameters and achieve 7-13% WERR improvement.

Domain prompts are deployment friendly: In addition to performance and cost benefits, *domain-prompts* can easily be used for new domains without having to deploy new models or introducing new architectures. *Domain prompts* are prefixed to the input keeping all the base model parameters unchanged, while all other adaptation methods require updating the parameters inside the base model architecture.

Table 3. Qualitative examples: domain-adapted *gpt2* prefers hypothesis (green) over hypothesis with incorrect tokens (red)

hypothesis	**perplexity ↓**	
	vanilla *gpt2*	adapted *gpt2*
insurance		
i would like to retrieve my code	**172.5**	40.4
i would like to retrieve my quote	238.1	**21.4**
airlines		
what's the point to tell you for frequent flyer number	**313.5**	19.9
what's the points tally for frequent flyer number	598.9	**17.6**

Domain prompts provides hyper-parameter (k) to trade-off performance and cost: Comparing row 13, 14 and 16 in Table 2, we see that the performance of models improves as we increase k, although the improvements saturate. This provides ASR system developers a parameter to control cost as per their requirements and availability of domain data.

Initialization with common vocabulary token embeddings helps: Comparing rows 14 and 15, initializing ϕ_D with token embedding of most frequent domain words gives marginal improvements. Since these words are representative of the domain, they prove to be a useful starting point.

5. CONCLUSION

Domain prompts provides a scalable and parameter-efficient method to add domain information to Transformer based LMs. It saves storage and training cost without compromising performance. It also achieves the best performance for new domains with only handful of available examples. Rather than updating the base model parameters, the new parameters are added as prefixes to input, hence our method doesn't require model deployments per domain. Therefore, our method becomes an ideal choice for on-the-fly adaptation of second pass LMs for incrementally scaling the industrial ASR system to new domains with negligible overhead.

6. REFERENCES

[1] Alex Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.

[2] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 4960–4964.

[3] Qian Zhang, Han Lu, Hasim Sak, Anshuman Tripathi, Erik McDermott, Stephen Koo, and Shankar Kumar, "Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7829–7833.

[4] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al., "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, pp. 9, 2019.

[5] 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.

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

[7] Ashish Shenoy, Sravan Bodapati, Monica Sunkara, Srikanth Ronanki, and Katrin Kirchhoff, "Adapting Long Context NLM for ASR Rescoring in Conversational Agents," in *Proc. Interspeech 2021*, 2021, pp. 3246–3250.

[8] Ashish Shenoy, Sravan Bodapati, and Katrin Kirchhoff, "Asr adaptation for e-commerce chatbots using cross-utterance context and multi-task language modeling," *Proceedings of The 4th Workshop on e-Commerce and NLP*, 2021.

[9] Nilaksh Das, Duen Horng Chau, Monica Sunkara, Sravan Bodapati, Dhanush Bekal, and Katrin Kirchhoff, "Listen, know and spell: Knowledge-infused subword modeling for improving asr performance of oov named entities," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 7887–7891.

[10] Dhanush Bekal, Ashish Shenoy, Monica Sunkara, Sravan Bodapati, and Katrin Kirchhoff, "Remember the context! asr slot error correction through memorization," in *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2021, pp. 236–243.

[11] Mahaveer Jain, Gil Keren, Jay Mahadeokar, Geoffrey Zweig, Florian Metze, and Yatharth Saraf, "Contextual rnn-t for open domain asr," *arXiv preprint arXiv:2006.03411*, 2020.

[12] Junho Park, Xunying Liu, Mark JF Gales, and Phil C Woodland, "Improved neural network based language modelling and adaptation," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[13] Tanel Alumäe, "Multi-domain neural network language model.," in *INTERSPEECH*, 2013, vol. 13, pp. 2182–2186.

[14] Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych, "Adapterhub: A framework for adapting transformers," *arXiv preprint arXiv:2007.07779*, 2020.

[15] Brian Lester, Rami Al-Rfou, and Noah Constant, "The power of scale for parameter-efficient prompt tuning," *arXiv preprint arXiv:2104.08691*, 2021.

[16] Xiang Lisa Li and Percy Liang, "Prefix-tuning: Optimizing continuous prompts for generation," *arXiv preprint arXiv:2101.00190*, 2021.

[17] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al., "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[18] Saket Dingliwal, Ashish Shenoy, Sravan Bodapati, Ankur Gandhe, Ravi Teja Gadde, and Katrin Kirchhoff, "Efficient domain adaptation of language models in ASR systems using prompt-tuning," *CoRR*, vol. abs/2110.06502, 2021.

[19] Reinhard Kneser and Hermann Ney, "Improved backing-off for m-gram language modeling," in *1995 international conference on acoustics, speech, and signal processing*. IEEE, 1995, vol. 1, pp. 181–184.

[20] Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan, "Dialogpt: Large-scale generative pre-training for conversational response generation," *arXiv preprint arXiv:1911.00536*, 2019.

[21] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher, "Pointer sentinel mixture models," *arXiv preprint arXiv:1609.07843*, 2016.