N-gram Is Back: Residual Learning of Neural Text Generation with n-gram Language Model

Huayang Li[♣] **Deng Cai**[♡] **Jin Xu**[♢] **Taro Watanabe**[♣]
[♣]Nara Institute of Science and Technology [♡]The Chinese University of Hong Kong
[♢]Institute for Interdisciplinary Information Sciences, Tsinghua University
{li.huayang.lh6, taro}@is.naist.jp thisisjcykcd@gmail.com
xujin21@mails.tsinghua.edu.cn

Abstract

N-gram language models (LM) have been largely superseded by neural LMs as the latter exhibits better performance. However, we find that n-gram models can achieve satisfactory performance on a large proportion of testing cases, indicating they have already captured abundant knowledge of the language with relatively low computational cost. With this observation, we propose to learn a neural LM that fits the residual between an n-gram LM and the real-data distribution. The combination of n-gram and neural LMs not only allows the neural part to focus on the deeper understanding of language but also provides a flexible way to customize an LM by switching the underlying n-gram model without changing the neural model. Experimental results on three typical language tasks (i.e., language modeling, machine translation, and summarization) demonstrate that our approach attains additional performance gains over popular standalone neural models consistently. We also show that our approach allows for effective domain adaptation by simply switching to a domain-specific n-gram model, without any extra training. Our code is released at https://github.com/ghrua/NgramRes.

1 Introduction

N-gram language model (LM) was widely adopted in a broad range of natural language processing (NLP) applications, such as input method (Chen et al., 2019), statistical machine translation (Brown et al., 1990), and audio speech recognition (Bahl et al., 1983). However, with the development of deep learning, neural LMs have gradually taken the place of n-gram LMs and became the new standard in recent literature (Merity et al., 2017; Vaswani et al., 2017; Radford et al., 2019). One critical reason is the superior performance of neural LMs, e.g., the GPT-2 model (Radford et al., 2019) can generate text near the human level, outperforming n-gram LMs by large margins.

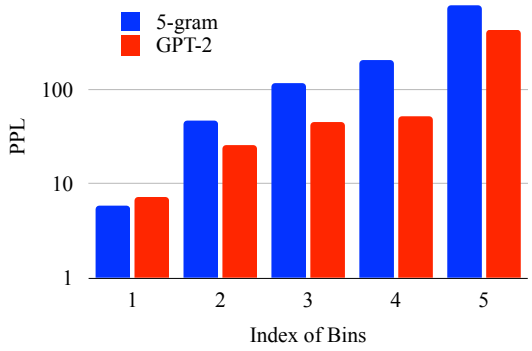

Figure 1: Sentence-level perplexity (PPL) of 5-gram LM and GPT-2 LM on the validation dataset of wikitext-103. We sort sentences in the validation dataset according to their 5-gram PPL scores, and collect them into 5 bins with an equal number of sentences. The reported PPL score of each bin is the average over the sentences in it, and the y-axis uses a logarithmic scale. Details of the dataset and LMs are shown in section 5.1.

Despite that neural LMs have surpassed n-gram models at the macro level, we find that n-gram LMs are still attractive: they are able to achieve satisfactory performance on a large proportion of testing cases at a much lower cost than neural LMs. As observed in Figure 1, our preliminary experiments show that the performance of 5-gram LM is close to the GPT-2 model trained from scratch on 3 out of 5 bins (1, 2, and 5). Moreover, the performance of 5-gram on the first bin is slightly better than GPT-2. Because training a neural LM is much more expensive, spending effort on learning the knowledge that can be cheaply captured by n-gram seems a waste.

Inspired by the above observation, we propose to learn a neural LM that focuses on the information gap that has not been captured by an n-gram model: $\mathcal{F} := \mathcal{G} - \mathcal{Q}$, where $\mathcal{G}$ and $\mathcal{Q}$ are the real-data distribution and the n-gram prediction distribution respectively, which is in a similar spirit to residual learning (He et al., 2016). More concretely, we combine the logits (the unnormalized probabil-

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 1523–1533
December 7-11, 2022

ity scores before `softmax` layer) of a neural model and those derived from an n-gram model. The joint neuro-symbolic system at least brings two appealing characteristics. First, since the neural model stands on the shoulders of the shallow n-gram LM, it can concentrate on deeper understanding. Second, the underlying n-gram LM can be purposefully switched without changing the neural model, which offers great flexibility in tackling scenarios such as domain adaptation. That is, we can adapt the model to a specific domain by changing the underlying n-gram LM in a plug-and-play manner, without changing any parameters of the neural model.

We conduct extensive experiments to evaluate the proposed approach. Experiments on the standard benchmarks of three typical language tasks, including language modeling, machine translation, and summarization, show that our approach can improve the performance of recent state-of-the-art neural models consistently and considerably. For example, our approach outperforms popular baseline models by at least 0.7 PPL scores on the wikitext-103 dataset for language modeling, 0.65 BLEU scores on average on IWSLT datasets for machine translation, and 0.36 ROUGE-L scores on the CNN/DailyMail dataset for summarization. Moreover, on the language modeling task, when switching the underlying n-gram LM to a particular domain-specific one (e.g., IT, Koran, Law, Medical, and Subtitles) in a plug-and-play manner, our model can reduce the PPL by 5.4 points on average without any domain-specific training of the neural part. Remarkably, the performance of our approach is even close to fine-tuning the whole model on domain-specific corpora.

Our contributions are three-fold:

- We propose a residual learning approach for two heterogeneous structures, i.e., n-gram and neural LMs, which forces the neural LM to approximate the information gap that has not been captured by n-gram LM.
- Our approach is able to improve the performance of recent state-of-the-art neural models consistently and considerably on language modeling, machine translation, and summarization.
- Experiments on domain adaptation demonstrate that our approach can effectively and cheaply adapt the model to a specific domain by changing the used n-gram LM in a plug-and-play manner, without changing any parameters of the neural model.

2 Related Work

Language Model The n-gram language model (LM) has been widely used in lots of applications of natural language processing (NLP) since a long time ago (Jurafsky, 2000). The emergence of advanced smoothing technologies makes the n-gram model able to provide a better estimation of human languages (Kneser and Ney, 1995; Chen and Goodman, 1996; Heafield et al., 2013). In statistical machine translation (Brown et al., 1990) and automatic speech recognition (Bahl et al., 1983), the decoder-side n-gram model is critical to estimate the quality of generated candidates. In recent literature on input methods, the n-gram LM is still the most popular choice for providing word suggestions (Huang et al., 2015; Chen et al., 2019), because of its low cost and low latency.

However, with the development of deep neural networks, the macro-level performance of neural LM has surpassed that of n-gram LM by a large margin. Comparing with the n-gram LM, one big advantage of the neural LM basing on recurrent neural network (Hochreiter and Schmidhuber, 1997; Chung et al., 2014) and attention neural network (Vaswani et al., 2017; Radford et al., 2019) is their ability to modeling long-distance dependencies (Grave et al., 2017). The success of neural LM can also be observed in the big improvement achieved in lots of downstream tasks, e.g., text generation (Holtzman et al., 2020; Welleck et al., 2020; Su et al., 2022; Xu et al., 2022; Li et al., 2022; Cai et al., 2022), machine translation (Bahdanau et al., 2015; Luong and Manning, 2015; Vaswani et al., 2017; Cai et al., 2021) and summarization (Li et al., 2017; See et al., 2017; Bi et al., 2020).

Although neural LM has outperformed n-gram LM at the macro level, we find that n-gram LM can achieve satisfactory performance on a large portion of testing cases. Since the training cost of neural LM is much more expensive and the model capacity is fixed, we hypothesize that it is not necessary to train the neural LM to learn the knowledge that can be captured by n-gram LM at a much lower cost. Therefore, we propose a residual learning method to let the neural LM learn the gap of knowledge that has not been captured by n-gram LM.

Residual Learning Residual learning is a useful technique for lots of neural networks in computer vision (CV) and natural language processing (NLP). He et al. (2016) propose deep residual learning to alleviate the training difficulties of deep models, which has been the backbone of lots of tasks in CV. In NLP, Wang and Tian (2016) and Prakash et al. (2016) use the residual learning technique to train deep recurrent neural networks for text generation. Different from previous works that conduct residual learning over different layers, Werlen et al. (2018) propose to aggregate the information of historical predictions using residual learning. In He et al. (2021), they use the residual learning to propagate attention scores across different layers of the Transformer-based model.

Most of these works conduct residual learning over homogeneous model structures, e.g., stacked identical layers of the same model. In our work, we use residual learning to combine the neural and symbolic models, i.e., learn a neural LM that approximates the information that has not been captured by the n-gram model.

3 Background

Models that estimate the probabilities of sequences of words are called language models (LM) (Jurafsky, 2000). Let $\boldsymbol{x} = \{x_1, x_2, ..., x_L\}$ be a sequence of words with length L. The probability of $P(\boldsymbol{x})$ can be formalized according to the chain rule of probability:

$$\begin{aligned} P(\boldsymbol{x}) &= P(x_1)P(x_2|x_1)\dots P(x_L|\boldsymbol{x}_1^{L-1}) \\ &= \prod_{k=1}^{L} P(x_k|\boldsymbol{x}_1^{k-1}), \end{aligned} \tag{1}$$

where $\boldsymbol{x}_1^{k-1}$ is called the prefix or context of x_k. In this section we will briefly introduce two kinds of language models, the n-gram and neural language models, to compute the probability in Eq. (1).

3.1 N-gram Language Model

Among lots of variants of n-gram LMs, the n-gram LM with modified Kneser-Ney smoothing is widely adopted in lots of related tasks, because of its low perplexity and efficiency (Kneser and Ney, 1995; Chen and Goodman, 1996; Heafield et al., 2013). Like most n-gram LMs, the Kneser-Ney approximates the entire context x_1^{k-1} in Eq. (1) by the last $n-1$ words in the context:

$$P(x_k|\boldsymbol{x}_1^{k-1}) \approx P_{NG}(x_k|\boldsymbol{x}_{k-n+1}^{k-1}). \tag{2}$$

In Kneser-Ney algorithm, the estimation of $P_{NG}(x_k|\boldsymbol{x}_{k-n+1}^{k-1})$ is defined according to a recursive equation:

$$\begin{aligned} P_{NG}(x_k|\boldsymbol{x}_{k-n+1}^{k-1}) = U(x_k|\boldsymbol{x}_{k-n+1}^{k-1}) + \\ b(\boldsymbol{x}_{k-n+1}^{k-1})P_{NG}(x_k|\boldsymbol{x}_{k-n+2}^{k-1}), \end{aligned} \tag{3}$$

$$U(x_k|\boldsymbol{x}_{k-n+1}^{k-1}) = \frac{c(\boldsymbol{x}_{k-n+1}^{k}) - d}{\sum_w c(\boldsymbol{x}_{k-n+1}^{k-1}w)},$$

where w indicates a word appears after $\boldsymbol{x}_{k-n+1}^{k-1}$, $b(\cdot)$ is the backoff value for lower-order estimation, $c(\cdot)$ is the adjusted counts, d is the discounts for smoothing (Jurafsky, 2000; Heafield et al., 2013)[1]. According to Eq. (3), Kneser-Ney allows us to assign probabilities for unseen n-grams (e.g., 5-grams), using the lower-order information (e.g., 4-, 3-, or even uni-grams).

3.2 Neural Language Model

An neural LM typically estimates the probability of x_k based on the whole context $\boldsymbol{x}_1^{k-1}$. The parameter θ of a neural LM is optimized through the following MLE loss:

$$\mathcal{L}_{NU} = \sum_{\boldsymbol{x}\in\mathcal{D}} \sum_{k=1}^{L} \log P_{NU}(x_k|\boldsymbol{x}_1^{k-1}; \theta) \tag{4}$$

where $\mathcal{D}$ is the training dataset. The probability of $P_{NU}(x_k|\cdot)$ is computed by:

$$P_{NU}(x_k|\boldsymbol{x}_1^{k-1}; \theta) = \mathrm{softmax}(\phi(\boldsymbol{h}_k))[x_k], \tag{5}$$

where $\boldsymbol{h}_k$ is the hidden vector output by the last layer of an neural LM, e.g., the GPT-2 model (Radford et al., 2019) or LSTM model (Grave et al., 2017). The $[x_k]$ is defined as taking the component regarding to x_k in a vector, i.e., the probabilistic distribution got from $\mathrm{softmax}$ in this equation. The $\phi(\cdot)$ is a linear layer that transforms the hidden vector $\boldsymbol{h}_k$ to a vector in the vocabulary space, which is also called the logits.

[1] More details about adjusting counts and computing the backoff values and discounts are shown in Jurafsky (2000) and Heafield et al. (2013).

4 Methodology

4.1 Motivation

The main idea of our work is to use the neural LM to approximate a residual function. Given the context $\boldsymbol{x}_1^{k-1}$ in the language modeling task, let us consider $\mathcal{G}(\boldsymbol{x}_1^{k-1})$ as the golden-truth distribution of the next word, and

$$\mathcal{Q}(\boldsymbol{x}_1^{k-1}) = P_{NG}(X|\boldsymbol{x}_{k-n+1}^{k-1}) \tag{6}$$

as the prediction distribution of the n-gram LM, where X is the random variable and the probability $P_{NG}(X = x_k|\boldsymbol{x}_{k-n+1}^{k-1})$ is calculated according to Eq. (3). Since the n-gram distribution $\mathcal{Q}(\boldsymbol{x}_1^{k-1})$ has captured abundant information of the language as we discussed in the introduction, one interesting question is: can we use a neural LM to approximate the residual function $\mathcal{F}(\boldsymbol{x}_1^{k-1}) := \mathcal{G}(\boldsymbol{x}_1^{k-1}) - \mathcal{Q}(\boldsymbol{x}_1^{k-1})$? This is similar to the residual learning in He et al. (2016). If it is possible, we can release the burden of neural LMs on learning the information that has been captured by n-gram LMs, e.g., short-distance dependencies, and provide a flexible way to customize an LM by switching the underlying n-gram model without changing the neural model.

4.2 Learning Objective

Ideally, to train a neural LM that approximates the residual function, one way is to re-define the $P_{NU}(x_k|\cdot)$ in Eq. (5) as follows:

$$P_{NU}(x_k|\boldsymbol{x}_1^{k-1};\theta) = \mathcal{F}(\boldsymbol{x}_1^{k-1})[x_k] + P_{NG}(x_k|\boldsymbol{x}_{k-n+1}^{k-1}),$$

where $\mathcal{F}(\cdot)$ is parameterized by the neural model θ, and $P_{NG}(x_k|\cdot)$ is defined in Eq. (3). Then we can optimize the MLE loss in Eq. (4) based on the new $P_{NU}(x_k|\cdot)$, which is equivalent to approximate real-data distribution $\mathcal{G}$ by $\mathcal{F} + \mathcal{Q}$. However, directly optimizing this objective may have some problems. If $\mathcal{F}(\cdot)$ is unbounded, P_{NU} defined in this equation may not be guaranteed as a valid probabilistic distribution. In contrast, if $\mathcal{F}(\cdot)$ is bounded as a valid distribution, this objective would become the ensemble of a neural LM and n-gram LM. Since n-gram is a weaker model, the ensemble of them is more likely to achieve worse performance than the vanilla neural LM, as shown in the experimental results of section 5.1.

To address these issues, we propose to define residual approximation at the logits level. In the language modeling task, we can map the probabilistic distribution back to its logits and conduct residual learning as follows:

$$\mathcal{F}'(\boldsymbol{x}_1^{k-1}) := \mathrm{softmax}^{-1}\big(\mathcal{G}(\boldsymbol{x}_1^{k-1})\big) - \mathrm{softmax}^{-1}\big(\mathcal{Q}(\boldsymbol{x}_1^{k-1})\big) \tag{7}$$

$$\mathrm{softmax}^{-1}(\boldsymbol{p}) = \log \boldsymbol{p} + C, \tag{8}$$

where $\mathcal{F}'(\cdot)$ is the residual function at the logits level, $\mathrm{softmax}^{-1}(\boldsymbol{p})$ is the reverse function of $\mathrm{softmax}$ that maps the probabilistic distribution $\boldsymbol{p}$ to its logits, and C is a constant. One reason that we conduct residual learning at the logits level is that logits are highly correlated to the final distribution. Moreover, since the value of logits is in the real number space, training the neural LM becomes more tractable by making sure that its logits are close to $\mathcal{F}'(\boldsymbol{x}_1^{k-1})$. Therefore, the final $P_{NU}(x_k|\cdot)$ defined in our work is:

$$P_{NU}(x_k|\boldsymbol{x}_1^{k-1};\theta) = \mathrm{softmax}\Big(\mathcal{F}'(\boldsymbol{x}_1^{k-1}) + \alpha\times \mathrm{softmax}^{-1}\big(\mathcal{Q}(\boldsymbol{x}_1^{k-1})\big)\Big)[x_k] \tag{9}$$

where α is a hyper-parameter to control the smoothness of the logits of the n-gram distribution $\mathcal{Q}(\boldsymbol{x}_1^{k-1})$, and $\mathcal{F}'(\cdot)$ is approximated by the logits $\phi(\boldsymbol{h}_k)$ of a neural LM. We can use the definition in Eq. (9) to optimize the MLE loss in Eq. (4).

4.3 Relation to Re-weighting

To better understand our approach, we can dive into the details of Eq. (9). For simplicity, let us omit the condition $\boldsymbol{x}_1^{k-1}$ in this section:

$$P_{NU}(x_k|\cdot) = \mathrm{softmax}\Big(\phi(\boldsymbol{h}_k) + \alpha\times \big(\log P_{NG}(X|\cdot) + C\big)\Big)[x_k] \tag{10}$$

$$= \frac{\big(\mathrm{e}^C\big)^\alpha \big(\mathrm{e}^{\log P_{NG}(x_k|\cdot)}\big)^\alpha \mathrm{e}^{\phi(\boldsymbol{h}_k)[x_k]}}{Z}, \tag{11}$$

We apply the Eq. (6) and (8) to get the explicit form of logits of the n-gram LM in Eq.(10), and the definition of $\phi(\boldsymbol{h}_k)$ is the same as that in Eq. (5). In Eq. (11), we expand the $\mathrm{softmax}$ function, where Z is the normalization term. The numerator of Eq. (11) has three terms. The first term $(\mathrm{e}^C)^\alpha$ is a constant for all the logit values, which does not affect the distribution. The middle term $(\mathrm{e}^{\log P_{NG}(x_k|\cdot)})^\alpha$

actually equals to $P_{NG}(x_k|\cdot)^{\alpha}$, which makes it be like the weight of the the logits of neural LM, i.e., the last term $\mathrm{e}^{\phi(\boldsymbol{h}_k)[x_k]}$ in Eq. (11). When comparing with the vanilla neural LM, the golden-truth words are not equally important in the learning process of our approach. For golden-truth words that are well estimated by the n-gram LM, our approach would get high probabilities after $\mathrm{softmax}$, leading to a small loss value for the neural module. As a result, the neural model can spend more effort on difficult cases, such as predictions relying on long-distance dependencies, which are hard to be estimated by the n-gram LM.

4.4 Discussion

In this section, we propose a method to conduct residual learning between the neural and symbolic models, i.e., neural LM and n-gram LM. One of our expectations about the joint neuro-symbolic system is its better understanding of language. To evaluate this hypothesis, we can test our approach on standard language tasks, such as language modeling, machine translation, and summarization. The other expectation is the plug-and-lay property of our approach. For instance, if the testing data come from different domains, we can change the $\mathcal{Q}$ in Eq. (9) by simply switching the used n-gram model.

5 Experiments

In our work, we consider three kinds of natural language generation tasks: language modeling, machine translation, and summarization. For the language modeling task, we first evaluate the performance of our approach on the standard setting of the language modeling task. Then we turn to a domain adaptation setting.

5.1 Language Modeling

Setup We use the wikitext-103 benchmark[2] to evaluate the performance of our approach in the standard setting. The training set contains around 101M tokens. Following Merity et al. (2017), tokens with a frequency lower than 3 have been replaced by the special token <unk> in the training datasets, and the number of remaining unique words is around 260k. For wikitext-103, we will train models at both word and subword levels. The subword-level data is preprocessed using `subword-nmt`[3] (Sennrich et al., 2016), where the number of merge operation is set to 32k.

We use `fairseq`[4] (Ott et al., 2019) as the code base of our neural modules. We implement our approach on two popular neural language models, GPT-2 base (Radford et al., 2019) and Adaptive Input (ADP) (Baevski and Auli, 2019). For the ADP model, we follow the original hyper-parameters and use the code released by Baevski and Auli (2019) in `fairseq`[5] to train the model on word-level data. Since the vocabulary size of the word-level data is too large, we train the GPT-2 base model on the subword-level data. For those neural models, we mostly use their default hyper-parameters reported in their paper (Baevski and Auli, 2019; Radford et al., 2019) and train those models from random initialization. Regarding to the n-gram model, we use the `KenLM`[6] (Heafield, 2011) to train n-gram models on both the word-level and subword-level data of wikitext-103. The n is set to 5 in our work. To make the perplexities of different models comparable, we report all the perplexity scores at the word level. For subword-level data, the word-level probability is the product of its subword tokens, following Baevski and Auli (2019).

When training our approach NGRAMRES, we will hybrid the KENLM-5GRAM model and the neural model, i.e., GPT-2 and ADP, using the residual learning method discussed in section 4. The hyper-parameter α in Eq. (9) is tuned according to the performance on the validation dataset.

Results As shown in Table 2, we evaluate our approach on the wikitext-103 benchmark. Although the macro performance of KENLM-5GRAM (Line 6) on the test set is poor, it is still able to promote the performance of our approach. When comparing our approach (Line 8 and 11) with the vanilla neural models (Line 7 and 9), our approach steadily outperforms ADP-FAIRSEQ[7] and GPT-2 by 0.7 and 0.9 PPL scores, respectively. According to these results, NGRAMRES is able to improve the model performance without changing the architecture and the number of parameters.

[2] Dataset provided by fairseq: https://s3.amazonaws.com/research.metamind.io/wikitext/wikitext-103-v1.zip

[3] https://github.com/rsennrich/subword-nmt

[4] https://github.com/facebookresearch/fairseq

[5] https://github.com/facebookresearch/fairseq/blob/main/examples/language_model/README.adaptive_inputs.md

[6] https://github.com/kpu/kenlm

[7] This is the result by running the officially released code of ADP

#		IT	Koran	Law	Medical	Subtitles	AVG.
1	#SENT	222,927	17,982	467,309	248,099	500,000	–
2	#WORD	2,585,965	4,512,266	15,348,052	4,512,266	5,125,239	–
3	KENLM-5GRAM	95.89	35.51	15.74	24.00	101.99	54.63
4	GPT-2	66.49	35.34	9.93	15.18	77.34	40.86
5	+ FINETUNE	53.69	26.77	9.43	12.96	69.33	34.44
6	+ NGRAMRES	54.29	28.08	8.93	13.29	71.80	35.28

Table 1: Test perplexity of five domains. Results in lines 1-2 are the statistical information of each domain. Results in lines 3-6 are the perplexity scores of different approaches when testing on the five domains. The GPT-2 and NGRAMRES (Line 4 and 6) approaches only train unified models for five domains, while the FINETUNE method (Line 5) trains a domain-specific model for each domain.

#	Model	#Param	PPL
1	(Grave et al., 2017) - LSTM	–	40.8
2	(Dauphin et al., 2017) - GCNN-8	229M	37.2
3	(Merity et al., 2018) - QRNN	151M	33.0
4	(Rae et al., 2018) - HEBBIAN + CACHE	–	29.2
5	(Baevski and Auli, 2019) - ADP	247M	18.7
6	KENLM-5GRAM	–	116.4
7	ADP-FAIRSEQ	247M	18.9
8	+ NGRAMRES	247M	18.2
9	GPT-2 (BPE)	185M	22.2
10	+ PROB-INTER	185M	60.2
11	+ NGRAMRES	185M	21.3

Table 2: Test perplexity on wikitext-103. Results in lines 1-5 are reported in previous works, and results in lines 6-11 are run by us. The NGRAMRES is our approach discussed in section 4.

Moreover, we also compare our method with a straightforward baseline PROB-INTER, as discussed in section 4. The PROB-INTER baseline directly interpolates the probabilistic distribution of KENLM-5GRAM and GPT-2. The performance of PROB-INTER is better than the KENLM-5GRAM but worse than the vanilla GPT-2, making it like the ensemble of the two models, as we discussed in the section 4.

5.2 Language Modeling: Multi-Domain

In this setting, we will evaluate the performance of adapting our approach to a specific domain by changing the used n-gram model.

Setup In the multi-domain setting, we use the English side of a bilingual dataset with 5 domains (Aharoni and Goldberg, 2020), i.e., IT, Koran, Law, Medical, and Subtitles. The statistical information of this dataset is shown in Table 1. we apply `subword-nmt` on the joint training data of five domains, and the number of the merge operation is also 32k.

Following the standard setting of the language modeling task, we use GPT-2 base (Radford et al., 2019) as the neural model. We train and select GPT-2 model on the mixed data from five domains, and report the word-level perplexity on the test data of each domain independently. The GPT-2 + FINETUNE method will adapt the parameters of GPT-2 model on the corresponding domain before testing. For our approach NGRAMRES, we train a 5-gram LM for each specific domain and switch the used 5-gram model to the corresponding domain during training and testing. It is worth noting that the neural parameters of NGRAMRES are fixed when testing.

Results The experimental results are shown in Table 1. For GPT-2 and NGRAMRES (Line 4 and 6), we train unified neural models on mixed data of five domains and evaluate their performances on the test data of five domains one by one. Results show that our approach can outperform the vanilla neural model GPT-2 by a large margin. Since the NGRAMRES approach stores a lot of domain-specific information in the 5-gram LM, we hypothesize that the neural module is able to learn useful and complementary knowledge during training, leading to the performance gain.

In the line of + FINETUNE, we also report the results of fine-tuning the GPT-2 model on each testing domain. It surprised us that the performances of our approach are very close to those of the FINETUNE method. The NGRAMRES even outperforms FINETUNE slightly on the Law domain. Moreover, compared with the FINETUNE, one advantage of our approach is its low cost of adapting our model to the testing domain, since we only need to replace the used 5-gram model in a plug-and-play manner.

Model	En ⇒ Fr	En ⇒ Es	En ⇒ Vi	En ⇒ De	AVG.
TRANSFORMER	39.96	36.99	28.55	27.79	33.32
+ NGRAMRES	40.27	37.27	29.60	28.05	33.79
+ NGRAMRES-ANNEAL	40.49	37.07	29.92	28.41	33.97

Table 3: BLEU scores on IWSLT. The TRANSFORMER model is the baseline, and NGRAMRES and NGRAMRES-ANNEAL are two variants of our approach. Comparing with NGRAMRES, the NGRAMRES-ANNEAL decreases the value of α in Eq. (9) linearly in the first 10k steps of model training.

Model	ROUGE-1	ROUGE-2	ROUGE-L
Pointer-generator + Coverage (See et al., 2017)	39.53	17.28	36.38
Mask Attention Network (Fan et al., 2021)	40.98	18.29	37.88
BertSum (Liu and Lapata, 2019)	42.13	19.60	39.18
UniLM (Dong et al., 2019)	43.08	20.43	40.34
UniLM V2 (Bao et al., 2020)	43.16	20.42	40.14
ERNIE-GEN-large (Xiao et al., 2021)	44.02	21.17	41.26
PEGASUS (Zhang et al., 2020)	44.17	21.47	41.11
ProphetNet (Qi et al., 2020)	44.20	21.17	41.30
PALM (Bi et al., 2020)	44.30	21.12	41.14
BART-LARGE (Lewis et al., 2020)	44.11	21.21	40.83
+ NGRAMRES	44.41	21.36	41.19

Table 4: ROUGE scores on the test set of CNN/DailyMail dataset.

5.3 Machine Translation

Next, we evaluate our approach on a popular sequence-to-sequence task, namely, machine translation. Note that we only integrate our approach into the decoder side of the encoder-decoder model.

Setup We conduct the experiments of machine translation on IWSLT14 (En ⇒ Fr, Es, De) and IWSLT15 (En ⇒ Vi). The IWSLT14 datasets[8] of three language pairs are preprocessed following the script provided by fairseq[9], where the evaluation data is sampled from the whole dataset and the test data is the concatenation of *dev2011*, *tst2012*, *tst2012*. There is no overlap between train, validation, and test sets. For IWSLT15, we use the train, evaluation, and test data preprocessed and released by Stanford[10] (Luong and Manning, 2015). The results are reported using tokenized SacreBLEU[11] (Post, 2018).

We use `fairseq` as our code base. We use the Transformer model as our architecture[12] for all the translation models. The Transformer model has 6 encoder layers and 6 decoder layers. Since the IWSLT datasets are small, the hidden size of FFN sublayers is set to 1024, the number of attention heads is set to 4, the dropout rate is set to 0.3, and the weight decay rate is set to 0.001. We set other hyper-parameters according to the default setting of Vaswani et al. (2017). All the translation models are trained for 30 epochs from random initialization.

The implementation details of the n-gram model and our approach are similar to that in the language modeling task. For the translation task, we only use the target data, i.e., the X side of En⇒X data, to train the KENLM-5GRAM LM.

Results The results of machine translation are shown in Table 3. We implement two variants of our approaches, namely, NGRAMRES and NGRAMRES-ANNEAL. The system of NGRAMRES only uses the 5-gram information on the decoder side, as we discussed in section 4. The difference between NGRAMRES and NGRAMRES-

[8] https://wit3.fbk.eu/2014-01

[9] https://github.com/facebookresearch/fairseq/blob/main/examples/translation/prepare-iwslt14.sh

[10] https://nlp.stanford.edu/projects/nmt/

[11] https://github.com/mjpost/sacrebleu

[12] The used architecture code in `fairseq` is `transformer_iwslt_de_en`

ANNEAL system is that the latter decreases the value of α linearly after each update . The alpha value becomes zero after 10k steps.

We find that both the two variants of our approaches outperform the TRANSFORMER model. The NGRAMRES-ANNEAL achieves the best results on each language pair, which means that the n-gram model is more critical for the beginning phase and may hurt the translation performance after that phase. According to Voita et al. (2021), the training of neural machine translation (NMT) systems undergoes three stages: target-side language modeling, learning the word-by-word translation, and learning to reorder. Therefore, we hypothesize that the use of the n-gram model in the whole training procedure may over-emphasize the importance of target-side language modeling in NMT, having a negative impact on the next two stages.

5.4 Abstractive Summarization

Lastly, we evaluate our approach on another popular sequence-to-sequence task, namely, abstractive summarization. Like machine translation, our approach is applied to the decoder side of the encoder-decoder model.

Setup For the abstractive summarization task, we preprocess the CNN/DailyMail dataset following the script provided by `fairseq`[13]. The evaluation metrics of the summarization task are ROUGE scores, i.e., ROUGE-1, ROUGE-2, and ROUGE-L (Lin, 2004)[14].

We follow the setting of previous works and fine-tune the pre-trained BART-LARGE model (Lewis et al., 2020) on the CNN/DailyMail dataset for 20k updates. We train the KENLM-5GRAM LM on the joint data of its source and summarization text.

Results The summarization task is also a sequence-to-sequence task, where the source text and summarization are in the same language and share similar semantics. As shown in Table 4, in this task, our approach is still able to improve the performance of the strong baseline model BART-LARGE, without any change in the model architecture.

Different from the machine translation task, we find that using a fixed α value achieves better performance than annealing it. The reason may be that the target-side language modeling plays a more important role in the summarization task because summarization is more like monolingual text generation in a constrained context.

[13] https://github.com/facebookresearch/fairseq/blob/main/examples/bart/README.summarization.md

[14] https://github.com/pltrdy/files2rouge

6 Conclusion and Future Work

This work aims to learn a neural LM that approximates the information that has not been captured by n-gram LM. To achieve this goal, we propose a residual learning approach to force the two neural and symbolic models, i.e., the neural LM and n-gram LM, to learn complementary information. We conduct extensive experiments to evaluate the performance of the proposed approach. In our experiments, we find that our neuro-symbolic system can not only improve the performance of recent state-of-the-art neural models consistently and considerable on three typical language tasks (including language modeling, machine translation, and summarization) but also exhibits a good plug-and-play property on the multi-domain language modeling task.

The n-gram LM has lots of attractive properties that we have not explored in this work. First, the n-gram model has good interpretability. The behavior of n-gram LM is easier to understand than the weights of neurons from the perspective of humans. In the future, we want to leverage the property of the n-gram model to better understand the decision-making process of the neural LM. Second, controlling the system predictions through the n-gram model may have a big potential. As observed in our multi-domain experiments, we are able to customize an LM by switching the underlying n-gram model without changing the neural part. It is also interesting to explore how to control the model output at a fine-grained level using the n-gram LM.

Limitations

We believe there are two limitations in our approach. First, since the estimation of the prediction distribution of n-gram models relies on CPU, the estimation speed by n-gram models may be slow when using a big batch size ($>> 8192*8$). Second, the performance gain of our current approach on high-resource datasets is not big. For instance, we also evaluate the performance of TRANSFORMER + NGRAMRES on WMT14 En-De (Vaswani et al., 2017), but the improvement is only 0.15 BLEU score. These limitations urge us to propose more efficient and effective approaches in future works.

Acknowledgement

We are particularly grateful for the help from Xiaojiang Liu, because this project would never have been conceived and completed without his generous and selfless support. We also want to thank the insightful discussions with Yixuan Su and the valuable comments from our anonymous reviewers, area chairs, and senior area chairs.

References

Roee Aharoni and Yoav Goldberg. 2020. Unsupervised domain clusters in pretrained language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7747–7763, Online. Association for Computational Linguistics.

Alexei Baevski and Michael Auli. 2019. Adaptive input representations for neural language modeling. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Lalit R. Bahl, Frederick Jelinek, and Robert L. Mercer. 1983. A maximum likelihood approach to continuous speech recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 5(2):179–190.

Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Jianfeng Gao, Songhao Piao, Ming Zhou, et al. 2020. Unilmv2: Pseudo-masked language models for unified language model pre-training. In *International Conference on Machine Learning*, pages 642–652. PMLR.

Bin Bi, Chenliang Li, Chen Wu, Ming Yan, Wei Wang, Songfang Huang, Fei Huang, and Luo Si. 2020. Palm: Pre-training an autoencoding&autoregressive language model for context-conditioned generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8681–8691.

Peter F. Brown, John Cocke, Stephen Della Pietra, Vincent J. Della Pietra, Frederick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Comput. Linguistics*, 16(2):79–85.

Deng Cai, Yan Wang, Huayang Li, Wai Lam, and Lemao Liu. 2021. Neural machine translation with monolingual translation memory. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7307–7318.

Deng Cai, Yan Wang, Lemao Liu, and Shuming Shi. 2022. Recent advances in retrieval-augmented text generation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3417–3419.

Mingqing Chen, Ananda Theertha Suresh, Rajiv Mathews, Adeline Wong, Cyril Allauzen, Françoise Beaufays, and Michael Riley. 2019. Federated learning of n-gram language models. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 121–130, Hong Kong, China. Association for Computational Linguistics.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *34th Annual Meeting of the Association for Computational Linguistics, 24-27 June 1996, University of California, Santa Cruz, California, USA, Proceedings*, pages 310–318. Morgan Kaufmann Publishers / ACL.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 933–941. PMLR.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. *Advances in Neural Information Processing Systems*, 32.

Zhihao Fan, Yeyun Gong, Dayiheng Liu, Zhongyu Wei, Siyuan Wang, Jian Jiao, Nan Duan, Ruofei Zhang, and Xuan-Jing Huang. 2021. Mask attention networks: Rethinking and strengthen transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1692–1701.

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2017. Improving neural language models with a continuous cache. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision*

and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, pages 770–778. IEEE Computer Society.

Ruining He, Anirudh Ravula, Bhargav Kanagal, and Joshua Ainslie. 2021. Realformer: Transformer likes residual attention. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 929–943. Association for Computational Linguistics.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland. Association for Computational Linguistics.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 690–696, Sofia, Bulgaria. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Guoping Huang, Jiajun Zhang, Yu Zhou, and Chengqing Zong. 2015. A new input method for human translators: Integrating machine translation effectively and imperceptibly. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1163–1169. AAAI Press.

Dan Jurafsky. 2000. *Speech & language processing*. Pearson Education India.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *1995 International Conference on Acoustics, Speech, and Signal Processing, ICASSP '95, Detroit, Michigan, USA, May 08-12, 1995*, pages 181–184. IEEE Computer Society.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Huayang Li, Yixuan Su, Deng Cai, Yan Wang, and Lemao Liu. 2022. A survey on retrieval-augmented text generation. *arXiv preprint arXiv:2202.01110*.

Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. 2017. Deep recurrent generative decoder for abstractive text summarization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2091–2100, Copenhagen, Denmark. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740.

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford neural machine translation systems for spoken language domain. In *International Workshop on Spoken Language Translation*, Da Nang, Vietnam.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. An analysis of neural language modeling at multiple scales. *CoRR*, abs/1803.08240.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Aaditya Prakash, Sadid A. Hasan, Kathy Lee, Vivek V. Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2016. Neural paraphrase generation with stacked residual LSTM networks. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 2923–2934. ACL.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. Prophetnet: Predicting future n-gram for sequence-to-sequencepre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2401–2410.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Jack W. Rae, Chris Dyer, Peter Dayan, and Timothy P. Lillicrap. 2018. Fast parametric learning with activation memorization. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4225–4234. PMLR.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Yixuan Su, Tian Lan, Yan Wang, Dani Yogatama, Lingpeng Kong, and Nigel Collier. 2022. A contrastive framework for neural text generation. *arXiv preprint arXiv:2202.06417*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Elena Voita, Rico Sennrich, and Ivan Titov. 2021. Language modeling, lexical translation, reordering: The training process of NMT through the lens of classical SMT. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8478–8491, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yiren Wang and Fei Tian. 2016. Recurrent residual learning for sequence classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 938–943. The Association for Computational Linguistics.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2020. Neural text generation with unlikelihood training. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Lesly Miculicich Werlen, Nikolaos Pappas, Dhananjay Ram, and Andrei Popescu-Belis. 2018. Self-attentive residual decoder for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1366–1379. Association for Computational Linguistics.

Dongling Xiao, Han Zhang, Yukun Li, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2021. Ernie-gen: an enhanced multi-flow pre-training and fine-tuning framework for natural language generation. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 3997–4003.

Jin Xu, Xiaojiang Liu, Jianhao Yan, Deng Cai, Huayang Li, and Jian Li. 2022. Learning to break the loop: Analyzing and mitigating repetitions for neural text generation. *CoRR*, abs/2206.02369.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.