MERGE: Fast Private Text Generation

Zi Liang, Pinghui Wang, Ruofei Zhang, Nuo Xu, Shuo Zhang,

MOE KLINNS Lab, Xi'an Jiaotong University, Xi'an 710049, P. R. China
{liangzid,zs412082986}@stu.xjtu.edu.cn, phwang@mail.xjtu.edu.cn,
rfzhang@gmail.com, nxu@sei.xjtu.edu.cn

Abstract

Recent years have seen increasing concerns about the private inference of NLP services and Transformer models. However, existing two-party privacy-preserving methods solely consider NLU scenarios, while the private inference of text generation such as translation, dialogue, and code completion remains unsolved. Besides, while migrated to NLG models, existing privacy-preserving methods perform poorly in terms of inference speed, and suffer from the convergence problem during the training stage. To address these issues, we propose MERGE, a fast private text generation framework for Transformer-based language models. Specifically, MERGE reuse the output hidden state as the word embedding to bypass the embedding computation, and reorganize the linear operations in the Transformer module to accelerate the forward procedure. Based on these two optimizations, extensive experiments show that MERGE can achieve a 26.5x speedup under the sequence length 512, and reduce 80% communication bytes, with an up to 10x speedup to existing state-of-art models.

1 Introduction

Recently, pre-trained language models (PLMs) based on Transformer Vaswani et al. [2017] have attracted significant attention because of their exceptional performance in downstream tasks. However, the deployment of such PLM-based services in real-world situations raises concerns about privacy. For example, existing NLP services like Copilot[1] and ChatGPT[2] require users to send their text queries to servers, where the information contained, such as source code, the medical information, and personal preferences, may be sensitive to users.

To alleviate the privacy problem, some of the recent works Hao et al. [2022], Chen et al. [2022] have developed 2-party secure inference services for PLMs by secure Multi-Party Computation (MPC). MPC ensures privacy by encrypting user data and model weights and sharing them secretly. However, PLMs inference under MPC is considerably slow compared to the plain-text version, which limits its application in real-world services. To address this issue, some works have attempted to simplify the bottleneck operation such as activation functions and softmax in the Transformer model. For instance, Mishra et al. [2020] uses Neural Architecture Search (NAS) to replace the activation functions with linear layers, while Li et al. [2022] approximates the exponential operation with polynomial functions.

Though designed for Transformer, these works Hao et al. [2022], Chen et al. [2022], Li et al. [2022] solely explore the scenario of NLU inference (e.g. GLUE Wang et al. [2019]), and our experiments suggest that they have **no** significant improvements in text generation tasks (Figure 1). For example, the existing framework MPCformer Li et al. [2022] achieves XXXx speedup to BERT-base but only

[1] `https://github.com/features/copilot`
[2] `https://chat.openai.com`

Preprint. Under review.

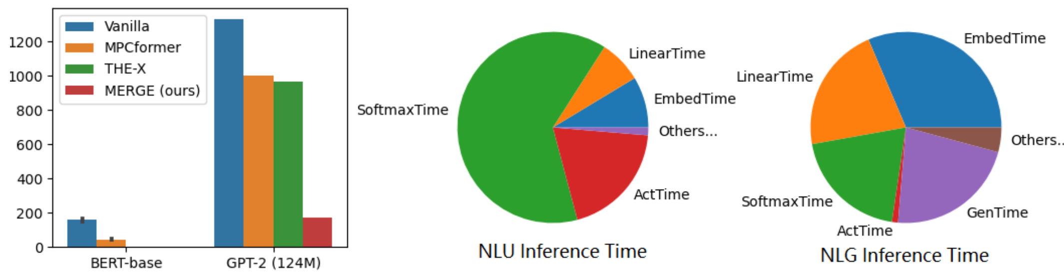

Figure 1: Inference Time Comparison among NLU and NLG models.

XXXXx for GPT-2 in text generation. By illustrating the inference bottleneck of NLU and NLG inference procedure, our experiments show that auto-regressive generation suffers from extra time cost in *embedding table query* and *token sampling* (i.e. GenTime), which slows down the whole inference procedure heavily.

In this paper, we explore to accelerate the generation procedure of language models. To this end, we propose *MERGE*[3], a fast and easy-to-adopt framework for private text generation. *MERGE* is compatible with previous MPC-based works (e.g. MPCformer, THE-X, and IRON) and mainstream PLMs (e.g. GPT-2 Radford et al. [2019], T5 Raffel et al. [2020], and Bart Lewis et al. [2020]). In *MERGE*, we first put forward a strategy called **embedding resending**, which directly uses the output hidden state as the new input token embedding. Embedding resending helps to bypass the *embedding table query* operation and decouple the computation between *forward representation learning* and *next token sampling*. Besides, following the recent research Hassid et al. [2022] in attention mechanism, we approximate *self-attention* with *constant attention* matrices and merge tensor computations in the Transformer module before inference. These two methods are challenging because: 1) PLMs are usually sensitive to input embeddings, while there are some unavoidable errors in the generated embeddings; 2) constant attention in our **merge module** might hurt the performance of PLMs. To address these issues, we first propose an embedding alignment and augmentation task to enhance the robustness of PLMs to input embeddings. Besides, we employed a weighted distillation training task for approximation models, which allowed us to overcome the negative effects of constant attention. Our empirical experiments on popular text generation tasks such as E2E Dusek et al. [2018], Multiwoz 2.1 Eric et al. [2020], and DailyDialog Li et al. [2017] demonstrate the effectiveness of *MERGE*. Specifically, it can achieve a considerable speedup of 7.75x to GPT-2 and 10.89x to T5 under the sequence length 128, while maintaining an acceptable performance with losses in BERTscore Zhang et al. [2020], BARTscore Yuan et al. [2021], and Rouge-L Lin [2004] of only 0.02 (under 0.92), 0.14 (under -2.90), and 0.03 (under 0.44), respectively.

2 Related Work

Secure Multi-Party Computation. The goal of MPC is to enable private computations among multiple parties. In general, an MPC system may employ various secure techniques, including garbled circuits Yao [1986], Goldreich et al. [2019], fully homomorphic encryption (FHE) Gentry [2009], and homomorphic secret sharing (HSS) Boyle et al. [2016]. Thanks to the rich support of existing MPC methods, it is practicable to implement the private inference of Transformer models. Therefore, rather than building a new MPC system, this paper focuses on accelerating the private generation procedure for Transformer-based language models. As a result, our method *MERGE* can provide much faster text generation which will offer significant benefits to existing mainstream MPC implementations. We detail the MPC system used in this paper in Appendix A.

MPC-oriented Approximations. Although existing MPC techniques can provide secure inference for neural networks, they usually suffer from prohibitively high communication delays and computation costs. This is primarily due to the critical nonlinear operations within neural networks. Therefore, some works aim to approximate these bottleneck operations in neural networks. For instance, Chen et al. [2022] replaces the GeLU activation function in the Transformer with ReLU, and

[3]**M**PC-based **E**mbedding **R**esending **GE**neration with layer **MERGE**

Hao et al. [2022] reformulate the $Tanh(\cdot)$ function in GeLU based on optimized exponential operations. Besides, Mishra et al. [2020] approximates the ReLU function with linear layers, and thus it can replace the MPC method used for ReLU, i.e. the garbled circuits, with secret sharing and Beaver triples. Similarly, Li et al. [2022] approximates GeLU with ReLU and quadratic functions. For the softmax operation in the attention mechanism, Li et al. [2022] approximates it by $softmax(x) \approx \frac{ReLU(x)}{\sum ReLU(x)}$ or $softmax(x) \approx \frac{(x+c)^2}{\sum (x+c)^2}$. However, these approximations were designed for the "one-time" inference such as NLU models (e.g. BERT), and are not optimized for auto-regressive generative models (e.g. GPT-series) that execute the forward inference multiple times.

3 Preliminary

3.1 Text Generation with Language Models

The text generation task (e.g. dialogue) aims to generate the desired sequence y (e.g. the response of the chatbot) under the given prefix text p (e.g. the dialogue context) with the language model $p_\theta(y|p)$. Typically, existing language models usually generate y in an *auto-regressive* manner, i.e.

$$p(y|p) = \prod_{t=1} p(x_t^y|p, x_{<t}^y), \tag{1}$$

where the x_t^y denotes the t-th generated token of y, and $x_{<t}^y$ denotes the generated sequence of y at step t.

In Equation 1, if we denote the one-hot representation of $(p, x_{<t}^y)$ as $\mathbf{x}_t$ with text length N_t, then the generation procedure can be divided into the following three stages:

a) Embedding table query, i.e. $\mathbf{E}_t = f_e(\mathbf{x}_t)$, where $f_e(\mathbf{x}) : \mathbb{R}^{N_t \times V} \to \mathbb{R}^{N_t \times d}$ is the embedding layer that maps the V-length index representation into a d-dimension semantic space;

b) Representation learning, i.e. $\mathbf{h}_t^{n_l} = \mathbf{f}_{tr}(\mathbf{E}_t')$, where $\mathbf{f}_{tr} : \mathbb{R}^{N_t \times d} \to \mathbb{R}^{N_t \times d}$ is a n_l-layer transformer model, $\mathbf{h}_t^{n_l}$ is the output hidden state, and $\mathbf{E}_t'$ is the combination of positional embeddings, token embeddings $\mathbf{E}_t$, and others.

c) Next token sampling, i.e. $x_t^y \sim f_{cls}(\mathbf{h}_t^{n_l})[N_t]$, where $f_{cls}(\mathbf{h}_t^{n_l}) : \mathbb{R}^{N_t \times d} \to \mathbb{R}^{N_t \times V}$ is the linear head, $f_{cls}(\mathbf{h}_t^{n_l})[N_t]$ denote the N_t-th item of $f_{cls}(\mathbf{h}_t^{n_l})$, and $\sim$ denotes the sampling strategy (e.g. greedy search) in a generation.

3.2 Transformer Module

In Section 3.1 **b)** the Transformer model $\mathbf{f}_{tr}$ can be seen as a stack of transformer modules. Specifically, each transformer module $f_{tr}^n : \mathbb{R}^{N_t \times d} \to \mathbb{R}^{N_t \times d}$ consists of following three components:

a) Projection, i.e. $\mathbf{Q}^n, \mathbf{K}^n, \mathbf{V}^n = W_{Q^n}^T \mathbf{h}^{n-1}, W_{K^n}^T \mathbf{h}^{n-1}, W_{V^n}^T \mathbf{h}^{n-1}$, where $W_{Q^n}, W_{K^n}, W_{V^n} \in \mathbb{R}^{d \times (d/N_h) \times N_h}$ are N_h-head projection matrices. Particularly, $\mathbf{h}^0 = \mathbf{E}_t'$.

b) Attention[4], i.e. $\mathbf{x}_{att}^n = f_{ln}(f_{dr}(W_{d^n}^T \cdot (\text{Concat}(A^n \cdot \mathbf{V}^n)) + b_{d^n}) + \mathbf{h}^{n-1})$, where $A^n \in \mathbb{R}^{N_h \times N_t \times N_t}$ is the N_h-head attention matrix that can be calculated by $A = f_{dr}(softmax(\mathbf{Q}^n \cdot \mathbf{K}^{nT} / \sqrt{d_k}))$, $d_k = d/N_h$, $W_{d^n} \in \mathbb{R}^{d \times d}$ is the weight matrix, $b_{d^n} \in \mathbb{R}^d$ is the bias, f_{dr} denotes the dropout operation Srivastava et al. [2014], and f_{ln} is the layer normalization Ba et al. [2016] layer.

$$f_{lyn}(\mathbf{x}) = \frac{\mathbf{x} - E[\mathbf{x}]}{\sqrt{Var[\mathbf{x}] + \epsilon}} \odot \gamma + \beta, \tag{2}$$

in which ϵ is a tiny number, $\odot$ denotes the element-wise product, and $E[\mathbf{x}]$ and $Var[\mathbf{x}]$ denote the mean and variance of $\mathbf{x}$, respectively.

[4] Noted that there are some slight differences for cross attention, e.g. in cross attention **K** and **V** are calculated with the output hidden state of the encoder. While it has no impact on our method in section 4, we will simply discuss the situation of self-attention.

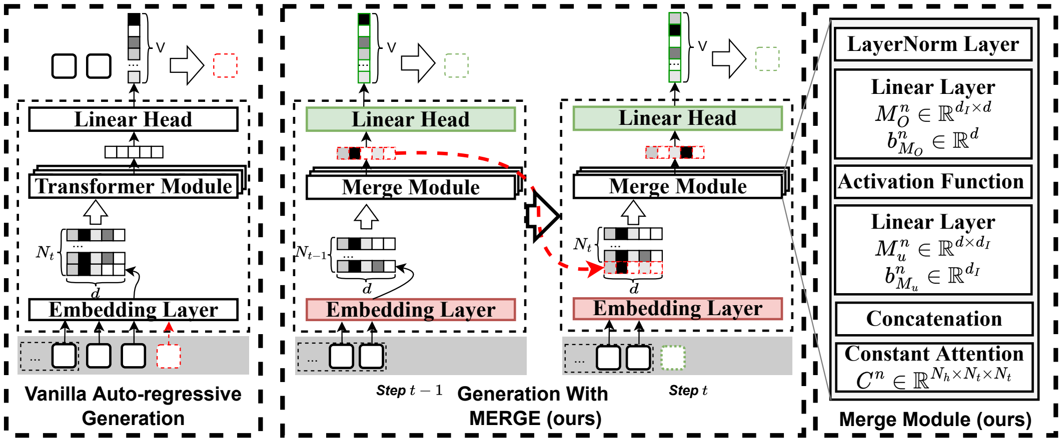

Figure 2: Generation Procedure and Architecture of MERGE.

c) Feed forward, i.e. $\mathbf{h}^n = f_{ln}(f_{dr}(W_O^{nT} \cdot (\text{Act}(W_I^{nT} \cdot \mathbf{x}_{att}^n + b_I^n) + b_O^n) + \mathbf{x}_{att}^n)$, where $W_I^n \in \mathbb{R}^{d \times d_I}$ and $W_O^n \in \mathbb{R}^{d_I \times d}$ are weighted matrices, $b_I^n \in \mathbb{R}^{d_I}$ and $b_O^n \in \mathbb{R}^d$ are bias vectors, d_I is the dimension of intermediate hidden states, and $\text{Act}(\cdot)$ denotes the activation functions such as ReLU Agarap [2018] and GeLU Hendrycks and Gimpel [2016].

4 MERGE

In this section, we present *MERGE*, a fast text generation framework for private inference. Illustrated in Figure 2, *MERGE* consists of two independent optimizations, the *embedding resending* (ER) strategy, and a new architecture built upon the *merge module* (MM).

4.1 Embedding Resending

As shown in Figure 2, the ER strategy aims to speed up the generation process by avoiding time-consuming operations (e.g. *embedding table query* in Section 3.1 **a)**) and decoupling the computation between *representation learning* (Section 3.1 **b)**) and *token sampling* (Section 3.1 **c)**). In detail, ER simply set the newly added token embedding $\mathbf{E}_t[N_t]$ as the generated hidden state at the last step ($\mathbf{h}_{t-1}^{n_l}[N_{t-1}]$), i.e.

$$\mathbf{E}_t = [\mathbf{E}_{t-1}; \mathbf{h}_{t-1}^{n_l}[N_{t-1}]] = [\mathbf{E}_0; \mathbf{h}_{t-1}^{n_l}], \tag{3}$$

where $\mathbf{E}_0$ denotes the token embeddings of the prefix p and ";" denotes the concatenation operation.

In intuition, Equation 3 regards *Embedding table query* (Section 3.1 **a)**) as the inverse procedure of *next token sampling* (Section 3.1 **c)**), which implies that hidden states and token embeddings are in the same representation space, and the embedding layer f_e is the inverse function of f_{cls}. Therefore, to align the representation between $\mathbf{h}_{t-1}^{n_l}[N_{t-1}]$ and $\mathbf{E}_t[N_t]$, we design a training task that maximizes the cosine similarity between these vectors, i.e.

$$\mathcal{L}_{cos} = \frac{1}{N_{tr} \cdot N} \sum_{i}^{N_{tr}} \sum_{t=1}^{N} 1 - cosine(\mathbf{h}_{i,t-1}^{n_l}[N_{t-1}], \mathbf{E}_{i,t}[N_t]), \tag{4}$$

where $cosine(a, b) = \frac{a \cdot b}{||a|| \cdot ||b||}$ is the cosine similarity, N_{tr} is the number of train set, and N denotes the sequence length.

In Equation 4 we select the cosine similarity instead of mean square error (MSE) because the inner product (e.g. *self-attention* in Section 3.2 **b)**) plays a key role in the Transformer module.

Besides, we observe that the error of token embeddings significantly impacts the performance of the Transformer model $\mathbf{f}_{tr}$ and leads to nonsensical sentence generation with the MSE value over 10^{-3}. To enhance the robustness of $\mathbf{f}_{tr}$, we introduce an *embedding augmentation* method that first masks each element e_t in $\mathbf{E}_t$ with a rate p, and then adds a uniform noise sampled from the small interval $(-\epsilon, \epsilon)$, i.e.

$$\tilde{e}_t = m_t \cdot (e_t + n_t), \tag{5}$$

where $m_t \sim$ Bernoulli$(1-p)$ and $n_t \sim$ Uniform$(-\epsilon, \epsilon)$.

Thus the cross-entropy loss can be formatted as

$$\mathcal{L}_{ce} = \frac{1}{N_{tr} \cdot N} \sum_{i}^{N_{tr}} \sum_{t=1}^{N} \mathbf{x}_t[N_t] \cdot \log f_{cls}(\mathbf{f}_{tr}(\tilde{\mathbf{E}}_t^{'}))[N_t], \tag{6}$$

where $\tilde{\mathbf{E}}_t^{'}$ is the combination of noised token embedding $\tilde{\mathbf{E}}_t$ and others.

The overall train loss can be formatted as

$$\mathcal{L} = \lambda \mathcal{L}_{cos} + (1-\lambda)\mathcal{L}_{ce}, \tag{7}$$

where $\lambda \in [0, 1]$ is the weighting factor.

4.2 Layer Merging

In this subsection, we focus on designing an efficient approximation of the Transformer module f_{tr} (Section 3.2), i.e. the merge module f_{mer}, to accelerate the inference in the *linear computation* and *softmax* function.

Following recent research Hassid et al. [2022], we first replace the dynamic self-attention matrix A^n with a constant attention matrix $C^n \in \mathbb{R}^{N_h \times N_t \times N_t}$. We initialize C^n with the average of A^n in train set, i.e.

$$C^n = \frac{1}{N_{tr}} \sum_{i}^{N_{tr}} A_i^n \tag{8}$$

Besides, we approximate the layer normalization f_{ln} in Section 3.2 **b)** with a simple element-wise multiplication $f'_{ln}(\mathbf{x}) = \mathbf{x} \odot \gamma + \beta$, inspired by the previous work Chen et al. [2022]. Consequently, the attention procedure presented in Section 3.2 **b)** can now be approximated as

$$\mathbf{x}_{att}^n = f'_{ln}(f_{dr}(W_d^{nT} \cdot (\text{Concat}(C^n \cdot \mathbf{V}^n)) + b_d^n) + \mathbf{h}^{n-1}). \tag{9}$$

Based on Equation 9, we can simplify the whole computation procedure by reorganizing matrix computations in f_{tr} and merging intermediate linear operations. Specifically, we can merge the projection operation W_V^n, the linear map W_d^n, the approximated layer normalization function f'_{ln}, as well as the first linear map in feed-forward W_I^n into a single linear layer, i.e. a weighted matrix $M_u^n \in \mathbb{R}^{d \times d_I}$ and a bias term $b_{M_u}^n \in \mathbb{R}^{d_I}$, which can be formatted as:

$$\begin{aligned} M_u^n &= (W_{V^n} \cdot W_d^n + \mathbf{1}) \odot \gamma \cdot W_I^n, \\ b_{M_u}^n &= W_I^{nT} \odot \gamma \cdot b_d^n + W_I^{nT} \cdot \beta + b_I^n, \end{aligned} \tag{10}$$

where $\mathbf{1} \in \mathbb{R}^{d \times d}$ is the residual term in attention module (Section 3.2 **b)**).

Equation 10 shows that there are **no** parameters dependent on input token embeddings $\mathbf{E}'_t$. Hence, we can compute M_u and b_{M_u} before the inference stage, thus reducing the computation during model execution. As a result, we can simplify the entire Transformer module into only three tensor multiplications, i.e.

$$\mathbf{x}_o^n = f_{mer}(\mathbf{h}^{n-1}) = f_{ln}(W_O^{nT} \cdot \text{Act}(M_u^{nT} \cdot C^n \cdot \mathbf{h}^{n-1} + b_{M_u}^n) + b_O^n) \tag{11}$$

Although it may appear possible to merge M_u^n with the previous linear matrix W_O^{n-1} in Equation 11 by approximating the layer normalization f_{ln} with f'_{ln}, we choose to keep them separate for the following two reasons. Firstly, the merged matrix $W_O^{n-1} \cdot M_u^n \in \mathbb{R}^{d_I \times d_I}$ has significantly more parameters than W_O plus M_u, since d_I is typically larger than d. Secondly, removing f_{ln} in Equation 11 will hurt the convergence of the merge module heavily during training (detailed in Section 5.4).

In addition, to derive Equation 10 and Equation 11, we need to swap W_v^n and C^n, which requires the verification that the matrix multiplications on the tensor $\mathbf{h}_t^{n-1}$ under different dimensions obeys the **commutative law**. Proofs of this assertion are available in Appendix B.

Model	Time/ Communication Time			Total Time	Speedup
	EmbedTime	LinearTime	SoftmaxTime		
		GPT2-base (124M)			
CrypTen	321.44/52.33	251.93/74.21	454.61/113.96	1328.26	1x
MPCformer (sm2relu)	316.75/51.55	253.57/76.56	181.14/45.59	1001.41	1.33x
MPCformer (sm2quad)	318.16/50.88	253.30/75.16	152.45/37.40	972.50	1.36x
THE-X	329.29/58.30	258.00/80.21	87.71/19.28	965.79	1.37x
MERGE (ours)	**5.17/0.87**	**157.50/53.97**	**0.00/0.00**	**171.38**	**7.75x**
MERGE (only ER)	5.41/0.95	260.36/80.00	477.76/124.83	834.13	1.59x
MERGE (only MM)	320.84/50.92	250.98/81.57	0.00/0.00	747.45	1.78x
		T5 (138M)			
CrypTen	323.46/53.36	328.09/96.08	693.73/175.57	1569.41	1x
MPCformer (sm2relu)	327.51/55.36	328.61/96.80	284.65/75.17	1207.63	1.30x
MPCformer (sm2quad)	324.81/52.03	325.97/92.89	235.54/58.47	1149.07	1.37x
THE-X	316.16/48.58	321.90/90.82	126.73/25.51	1050.28	1.49x
MERGE (ours)	**7.62/1.27**	**131.31/44.11**	**0.00/0.00**	**144.02**	**10.89x**
MERGE (only ER)	8.24/1.58	211.57/65.19	596.74/166.50	874.36	1.79x
MERGE (only MM)	322.38/51.35	221.57/69.22	0.00/0.00	693.30	2.26x

Table 1: Inference Time Comparison of Private Text Generation Models.

Model	EmbedBytes	LinearBytes	SoftmaxBytes	TotalBytes	Fraction
		GPT2-base (124M)			
CrypTen	71.41GB	159.36GB	1.62GB	322.54GB	100.00%
MPCformer (sm2relu)	71.41GB	135.54GB	0.54GB	317.20GB	98.34%
MPCformer (sm2quad)	71.41GB	135.54GB	0.07GB	316.73GB	98.20%
THE-X	71.41GB	135.54GB	0.50GB	319.14GB	98.95%
MERGE (ours)	**1.15GB**	**119.89GB**	**0.00GB**	**121.76GB**	**37.75%**
MERGE (only ER)	1.15GB	160.63GB	1.62GB	168.51GB	52.24%
MERGE (only MM)	71.41GB	119.89GB	0.00GB	281.88GB	87.39%
		T5 (138M)			
CrypTen	147.14GB	199.97GB	7.72GB	380.45GB	100.00%
MPCformer (sm2relu)	147.14GB	199.97GB	2.73GB	364.74GB	95.87%
MPCformer (sm2quad)	147.14GB	199.97GB	0.33GB	362.33GB	95.24%
THE-X	147.14GB	199.97GB	2.97GB	369.73GB	97.18%
MERGE (ours)	**1.73GB**	**95.66GB**	**0.00GB**	**98.03GB**	**25.77%**
MERGE (only ER)	1.73GB	120.17GB	7.56GB	132.44GB	34.81%
MERGE (only MM)	73.72GB	95.66GB	0.00GB	257.89GB	67.79%

Table 2: Averaged Communication Bytes for Private Text Generation.

5 Experiments

5.1 Settings

Datasets. We evaluate *MERGE* on three representative text generation tasks, including Multiwoz Eric et al. [2020], a human-human multi-turn task-oriented dialogue corpus, DailyDialog Li et al. [2017], a multi-turn chitchat dataset, and CommonGen Lin et al. [2020], a hard-constrained controlled text generation benchmark. The details of datasets can be seen in Appendix C.1.

Baselines. We compare *MERGE* with several state-of-the-art private inference models and frameworks, including:

• **THE-X** Chen et al. [2022], one of the first approximation architecture of Transformer models;

• **MPCformer** Li et al. [2022], the approximated model that aims to accelerate the inference procedure of Transformer;

• **Crypten** Knott et al. [2021], one of the MPC implementations for PyTorch.

Evaluation Metrics. We evaluate *MERGE* in two dimensions: inference speed, and the effectiveness of approximation models. For inference speed, we record both the computation time and the communication bytes for each method. For the effectiveness of PLMs, we use Meteor Banerjee and Lavie [2005], CHRF++ Popovic [2017], NIST Lin and Och [2004], ROUGE family Lin [2004], BERTscore Zhang et al. [2020], and BARTscore Yuan et al. [2021] as the metrics. A detailed introduction can be found in Appendix C.2.

5.2 Implementation Details

We use GPT-2 (124M) Radford et al. [2019], T5-small Raffel et al. [2020], and Bart-base Lewis et al. [2020] as the basic evaluation backbone, with max sequence length 128. We trained all models under the learning rate 3×10^{-5}, batch size 4 with 3 epochs, based on the implementation of huggingface Transformers Wolf et al. [2020]. As for the distillation of approximated models, we train our baselines under the same hyperparameter settings in their source code, and train *MERGE* with 50000 steps under the learning rate 8×10^{-5}. All experiments above are on a single 32 GB Nvidia Tesla V100 GPU. Following previous works Li et al. [2022], for the experiments of private inference, we use two 32 GB Nvidia Tesla V100 GPUs to simulate the client and the server, with 10 GbE Ethernet bandwidth. We implement the whole MPC system based on Crypten Knott et al. [2021], a semi-honest MPC framework built on PyTorch. The implementation detail can be seen in Appendix A.

5.3 Speed Evaluation

We evaluate the inference speed under two mainstream NLG architecture, i.e. the pure decoder represented by GPT-2, and the encoder-decoder models represented by T5. We evaluate these two architectures with the sequence length 128, and record the total inference time as well as the time cost of each operation. As shown in Table 1, our method *MERGE* can obtain a 7.75x speedup to the encrypted GPT-2, and 5.8x to MPCformer. Besides, the vanilla encrypted GPT-2 with our embedding resending (MERGE only ER) can obtain a 59x speedup on *embedding table query*, and our merge module can help GPT-2 and T5 reduce half of the linear inference time, and achieve zero time cost in the softmax of attentions. Another phenomenon is that MERGE achieves a higher speedup on T5 than GPT-2, which is because in T5 every self-attention module follows with a cross-attention module.

Under the same settings of Table 1, we also record the communication bytes between the client and the server, shown in table 2. We can see existing methods reduce the communication volume slightly (less than 2% in GPT-2), while our method can reduce 62% communication bytes, with 98% and 25% on *embedding table query* and *linear operation*, respectively.

5.4 Performance Evaluation

Based on the improvements of inference speed, we focus on the inference performance between our *MERGE* method and other MPC frameworks. Table 3 shows the effectiveness of our methods and baselines, where the BERTscore of our *MERGE* method is lower than MPCformer with ReLU approximation (MPCformer (sf2relu)) by 0.01?, 0.017, and 0.001 in MultiWoz, CommonGen, and DailyDialog, respectively. This demonstrates that our methods maintain the comparable results to these baselines. Besides, Table 3 indicates that some acceleration methods designed for NLU models are not suitable to text generation models, i.e. they suffer from the convergence problem during training. For instance, THE-X replaces all *layer normalization* operations to the approximate normalization, which we observed will lead to the **out of time (OOT)** issue. Similarly, MPCformer that replace the softmax function to quadratic functions (MPCformer (sf2quad)) faces the same problem, though we train it with an elaborate layer-wise knowledge distillation.

Model	BERTscore	BARTscore	NIST	Rouge-L	METEOR	CHRF++
			MultiWoz NLG Eric et al. [2020]			
GPT-2 (124M)	0.9237	-2.9020	4.7907	0.4424	0.4900	43.2777
+ER (no train)	0.6860	-5.0660	0.2325	0.0707	0.0425	3.9721
+MPCformer (sf2relu)	0.9287	-2.5377	5.7248	0.4806	0.5792	48.8241
+MPCformer (sf2quad)	OOT	OOT	OOT	OOT	OOT	OOT
+THE-X	OOT	OOT	OOT	OOT	OOT	OOT
+MERGE (Ours)	0.8984	-3.1464	3.7444	0.3970	0.4302	36.6983
+MERGE only ER	0.9155	-2.8057	5.0812	0.4339	0.5102	44.2484
+MERGE only MM	0.9268	-2.6277	5.6524	0.4778	0.5647	47.7262
T5-small (60M)	0.9140	-2.8916	4.245	0.4216	0.5225	45.0229
+ER (no train)	0.0	-5.0347	-	0.0	0.0	0.0
+MPCformer (sf2relu)	0.9126	-2.7133	4.2952	0.4053	0.5354	45.5565
+MPCformer (sf2quad)	OOT	OOT	OOT	OOT	OOT	OOT
+THE-X	OOT	OOT	OOT	OOT	OOT	OOT
+MERGE (ours)	-	-	-	-	-	-
+MERGE only ER	0.9053	-3.1444	4.3608	0.3789	0.4379	38.2502
+MERGE only MM	0.9123	-2.8744	4.6270	0.4176	0.4879	42.6995
Bart-base	0.9301	-2.5284	5.8325	0.4889	0.5823	49.1391
+ER (no train)	0.0491	-5.0379	-	0.0038	0.0009	0.0507
+MPCformer (sf2relu)	0.8318	-4.1432	1.3971	0.1956	0.2157	19.2337
+MPCformer (sf2quad)	OOT	OOT	OOT	OOT	OOT	OOT
+THE-X	OOT	OOT	OOT	OOT	OOT	OOT
+MERGE (ours)	-	-	-	-	-	-
+MERGE only ER	0.9305	-2.4158	6.8489	0.5329	0.6070	52.5836
+MERGE only MM	0.8868	-3.6204	3.5688	0.3022	0.3662	31.6465
			CommonGen Lin et al. [2020]			
GPT-2 (124M)	0.9336	-3.4710	3.7840	0.2744	0.3012	27.7038
+ER (no train)	0.5999	-4.9864	0.0701	0.0192	0.0066	0.9470
+MPCformer (sf2relu)	0.8943	-4.1436	2.1301	0.1861	0.2691	27.6167
+MPCformer (sf2quad)	OOT	OOT	OOT	OOT	OOT	OOT
+THE-X	OOT	OOT	OOT	OOT	OOT	OOT
+MERGE (ours)	0.8821	-4.2479	0.6639	0.2025	0.1538	16.0573
+MERGE only ER	0.8953	-3.8979	1.6796	0.2430	0.2110	20.8878
+MERGE only MM	0.9083	-4.0885	2.2687	0.2026	0.2058	20.9888
			DailyDialog Li et al. [2017]			
GPT-2 (124M)	0.8404	-6.6387	0.5429	0.1142	0.1042	11.5089
+ER (no train)	0.7518	-6.8820	0.1287	0.0566	0.0526	6.8067
+MPCformer (sf2relu)	0.8161	-6.3494	1.1102	0.1322	0.1261	12.0713
+MPCformer (sf2quad)	OOT	OOT	OOT	OOT	OOT	OOT
+THE-X	OOT	OOT	OOT	OOT	OOT	OOT
+MERGE (ours)	0.8213	-6.2384	0.3674	0.1233	0.0955	7.8091
+MERGE only ER	0.8205	-6.5515	0.1069	0.1301	0.0833	6.5819
+MERGE only MM	0.8343	-6.5800	1.0499	0.1525	0.1364	14.9039

Table 3: Performance Experiments of Private Text Generation.

6 Analysis

6.1 Varying Sequence Lengths and Model Parameters

In this section, we dive to explore the effectiveness of our *MERGE* method under longer sequence length and larger model parameters. For sequence length, we set it from 64 to 512, and record the averaged score as well as the minimum and maximum score for each point. Illustrated by Figure 3, we can see the inference time cost as well as the communication volume decreases with the improvements of sequence length. In detail, our *MERGE* method can obtain a 26.5x speedup to the vanilla model and 11.8x to existing state-of-the-art model THE-X under sequence length 512, and reduce almost 80% communication Bytes. Besides, we can see our embedding resending (ER) strategy can obtain a **constant** embedding inference time, which is because ER bypasses the *embedding table query*, and thus its embedding time only related to the generation prefix of samples.

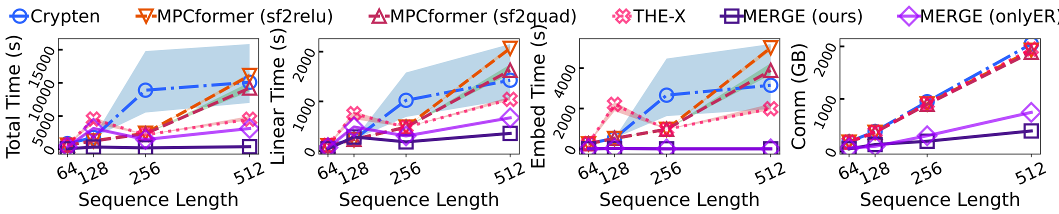

Figure 3: Experimental Results varying Model Parameters.

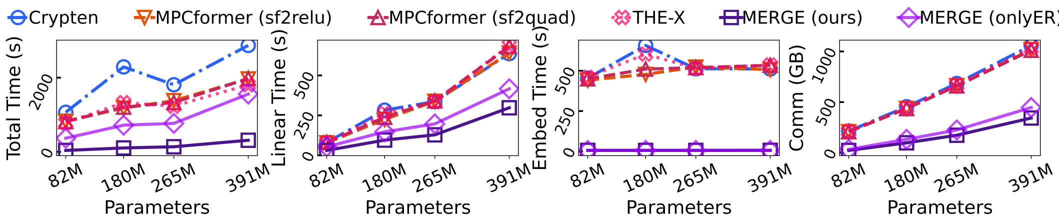

Figure 4: Experimental Results varying Model Parameters.

For model parameters, we also evaluate *MERGE* under different model sizes from 82M to 391M, and set the sequence length to 128. Different from Figure 3, Figure 4 demonstrates that there are no significant improvements of speedup while the model size increasing, but our *MERGE* method still obtains an obvious speedup (~10x) to existing methods. Besides, our method exhibits a conspicuous positive correlation with the model parameter size in terms of the gap between our method and the baselines, particularly in linear time and the communication volume, which demonstrate the effectiveness of *MERGE*.

7 Conclusion

In this paper, we address the problem of private text generation, and propose MERGE, a novel framework to accelerate the inference procedure of existing generative language models. MERGE consists of two optimizations, embedding resending and the merge module. The former speeds up the auto-regressive generation by bypassing the embedding table query of vanilla Transformer models, and the latter optimizes and merges the computation of Transformer modules. Extensive experiments demonstrate the superiority of our method both in inference speed and the generation quality. In the future, we plan to design a fast and plug-and-play MPC framework for existing language models.

References

Abien Fred Agarap. Deep learning using rectified linear units (relu). *CoRR*, abs/1803.08375, 2018. URL `http://arxiv.org/abs/1803.08375`.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Satanjeev Banerjee and Alon Lavie. METEOR: an automatic metric for MT evaluation with improved correlation with human judgments. In Jade Goldstein, Alon Lavie, Chin-Yew Lin, and Clare R. Voss, editors, *Proceedings of the Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization@ACL 2005, Ann Arbor, Michigan, USA, June 29, 2005*, pages 65–72. Association for Computational Linguistics, 2005. URL `https://aclanthology.org/W05-0909/`.

Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1292–1303, 2016.

Tianyu Chen, Hangbo Bao, Shaohan Huang, Li Dong, Binxing Jiao, Daxin Jiang, Haoyi Zhou, Jianxin Li, and Furu Wei. THE-X: privacy-preserving transformer inference with homomorphic encryption. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Findings of ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 3510–3520. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.findings-acl.277. URL `https://doi.org/10.18653/v1/2022.findings-acl.277`.

Ondrej Dusek, Jekaterina Novikova, and Verena Rieser. Findings of the E2E NLG challenge. In Emiel Krahmer, Albert Gatt, and Martijn Goudbeek, editors, *Proceedings of the 11th International Conference on Natural Language Generation, Tilburg University, The Netherlands, November 5-8, 2018*, pages 322–328. Association for Computational Linguistics, 2018. doi: 10.18653/v1/w18-6539. URL `https://doi.org/10.18653/v1/w18-6539`.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, Adarsh Kumar, Anuj Kumar Goyal, Peter Ku, and Dilek Hakkani-Tür. Multiwoz 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. In Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asunción Moreno, Jan Odijk, and Stelios Piperidis, editors, *LREC 2020, Marseille, France, May 11-16, 2020*, pages 422–428. European Language Resources Association, 2020. URL `https://aclanthology.org/2020.lrec-1.53/`.

Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *STOC 2009,*, pages 169–178. ACM, 2009. doi: 10.1145/1536414.1536440. URL `https://doi.org/10.1145/1536414.1536440`.

Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game, or a completeness theorem for protocols with honest majority. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 307–328. 2019.

Meng Hao, Hongwei Li, Hanxiao Chen, Pengzhi Xing, Guowen Xu, and Tianwei Zhang. Iron: Private inference on transformers. In *NeuIPS*, 2022.

Michael Hassid, Hao Peng, Daniel Rotem, Jungo Kasai, Ivan Montero, Noah A. Smith, and Roy Schwartz. How much does attention actually attend? questioning the importance of attention in pretrained transformers. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 1403–1416. Association for Computational Linguistics, 2022. URL `https://aclanthology.org/2022.findings-emnlp.101`.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Brian Knott, Shobha Venkataraman, Awni Y. Hannun, Shubho Sengupta, Mark Ibrahim, and Laurens van der Maaten. Crypten: Secure multi-party computation meets machine learning. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *NeurIPS 2021,*, pages 4961–4973, 2021. URL `https://proceedings.neurips.cc/paper/2021/hash/2754518221cfbc8d25c13a06a4cb8421-Abstract.html`.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault, editors, *ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.703. URL `https://doi.org/10.18653/v1/2020.acl-main.703`.

Dacheng Li, Rulin Shao, Hongyi Wang, Han Guo, Eric P. Xing, and Hao Zhang. Mpcformer: fast, performant and private transformer inference with MPC. *CoRR*, abs/2211.01452, 2022. doi: 10.48550/arXiv.2211.01452. URL `https://doi.org/10.48550/arXiv.2211.01452`.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. Dailydialog: A manually labelled multi-turn dialogue dataset. In Greg Kondrak and Taro Watanabe, editors, *IJCNLP 2017,* *Taipei, Taiwan, November 27 - December 1, 2017 - Volume 1: Long Papers*, pages 986–995. Asian Federation of Natural Language Processing, 2017. URL `https://aclanthology.org/I17-1099/`.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. Commongen: A constrained text generation challenge for generative commonsense reasoning. In Trevor Cohn, Yulan He, and Yang Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 1823–1840. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.findings-emnlp.165. URL `https://doi.org/10.18653/v1/2020.findings-emnlp.165`.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Annual Meeting of the Association for Computational Linguistics*, 2004.

Chin-Yew Lin and Franz Josef Och. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In Donia Scott, Walter Daelemans, and Marilyn A. Walker, editors, *ACL 2004, Barcelona, Spain*, pages 605–612. ACL, 2004. doi: 10.3115/1218955.1219032. URL `https://aclanthology.org/P04-1077/`.

Pratyush Mishra, Ryan Lehmkuhl, Akshayaram Srinivasan, Wenting Zheng, and Raluca Ada Popa. Delphi: A cryptographic inference service for neural networks. In Srdjan Capkun and Franziska Roesner, editors, *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*, pages 2505–2522. USENIX Association, 2020. URL `https://www.usenix.org/conference/usenixsecurity20/presentation/mishra`.

Maja Popovic. chrf++: words helping character n-grams. In Ondrej Bojar, Christian Buck, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno-Yepes, Philipp Koehn, and Julia Kreutzer, editors, *Proceedings of the Second Conference on Machine Translation, WMT 2017, Copenhagen, Denmark, September 7-8, 2017*, pages 612–618. Association for Computational Linguistics, 2017. doi: 10.18653/v1/w17-4770. URL `https://doi.org/10.18653/v1/w17-4770`.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020. URL `http://jmlr.org/papers/v21/20-074.html`.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1): 1929–1958, 2014. doi: 10.5555/2627435.2670313. URL `https://dl.acm.org/doi/10.5555/2627435.2670313`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *NeuIPS 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. URL `https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html`.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL `https://openreview.net/forum?id=rJ4km2R5t7`.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In Qun Liu and David Schlangen, editors, *EMNLP 2020 - Demos, Online,*, pages 38–45. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://doi.org/10.18653/v1/2020.emnlp-demos.6.

Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 162–167. IEEE Computer Society, 1986. doi: 10.1109/SFCS.1986.25. URL https://doi.org/10.1109/SFCS.1986.25.

Weizhe Yuan, Graham Neubig, and Pengfei Liu. Bartscore: Evaluating generated text as text generation. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *NeurIPS 2021, December 6-14, 2021, virtual*, pages 27263–27277, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/e4d2b6e6fdeca3e60e0f1a62fee3d9dd-Abstract.html.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with BERT. In *ICLR 2020,*. OpenReview.net, 2020. URL https://openreview.net/forum?id=SkeHuCVFDr.