GEP: A GCG-BASED METHOD FOR EXTRACTING PERSONALLY IDENTIFIABLE INFORMATION FROM CHATBOTS BUILT ON SMALL LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Small language models (SLMs) become unprecedentedly appealing due to their approximately equivalent performance compared to large language models (LLMs) in certain fields with less energy and time consumption during training and inference. However, the personally identifiable information (PII) leakage of SLMs for downstream tasks has yet to be explored. In this study, we investigate the PII leakage of the chatbot based on SLM. We first finetune a new chatbot, i.e., ChatBioGPT based on the backbone of BioGPT using medical datasets Alpaca and HealthCareMagic. It shows a matchable performance in BERTscore compared with previous studies of ChatDoctor and ChatGPT. Based on this model, we prove that the previous template-based PII attacking methods cannot effectively extract the PII in the dataset for leakage detection under the SLM condition. We then propose **GEP**, which is a greedy coordinate gradient-based (GCG) method specifically designed for PII extraction. We conduct experimental studies of GEP and the results show an increment of up to 60$\times$ more leakage compared with the previous template-based methods. We further expand the capability of GEP in the case of a more complicated and realistic situation by conducting free-style insertion where the inserted PII in the dataset is in the form of various syntactic expressions instead of fixed templates, and GEP is still able to reveal a PII leakage rate of up to 4.53%.

1 INTRODUCTION

LLM is one of the most centric research concentrations in the Artificial Intelligence (AI) field. It contributes dramatically to various domains (Zhao et al., 2023; Xu et al., 2024) and tasks (Zhao et al., 2023). Nevertheless, with the scaling up of parameters of LLMs, the energy and resource consumption are huge and unsustainable (Bolón-Canedo et al., 2024). Instead, SLM has gradually become the research focus in recent years. The idea is to make the models smaller, usually less than 7 billion parameters (Hu et al., 2024) but still with emergent ability (Wang et al., 2024), and to train them in a specific domain so that they can match the performance of LLMs in this certain field.

In spite of the protruding advantages of the SLMs, the privacy issues need to be considered before the practical deployment. Enormous amount of training data from the Internet may exhibit poor data quality and unintended leakage of private personal information (Das et al., 2025). Even data sanitation is unable to remove them completely (Carlini et al., 2019). These unexpected existences will leave chances for models to memorize some of them (Carlini et al., 2019), and suffer from revelation in the later inference phase. These private sensitive data are named as PII which includes the information such as person's name, telephone number and email address (Lukas et al., 2023). This is a common dilemma that almost all language models need to face. The inappropriate disclosure of PII will become a severe issue in many domains, such as the medical field as it does harm to patients both physically and mentally (Nakamura et al., 2020). Despite plenty of studies (Carlini et al., 2019; Nakamura et al., 2020; Lukas et al., 2023; Lehman et al., 2021; Huang et al., 2022; Chen et al., 2024; Nakka et al., 2024; Kim et al., 2023) that have already concentrated on the detection of potential PII leakage of different language models, there are fewer studies exploring the possibility of PII leakage of downstream tasks based on SLMs such as chat models. As the popularity of chat models starts to rise drastically in various fields (Dam et al., 2024) and they perform even better than

experts, relieving the burden of support staff (Ayers et al., 2023), it shows promising prowess of the chat model in practical scenarios greatly, as well as the urgency to protect the privacy which might be revealed by unreasonable design (Jain et al., 2023).

Another problem of previous studies is that the majority utilizes template-based sensitive data insertion or queries (Carlini et al., 2019; Lukas et al., 2023; Lehman et al., 2021; Huang et al., 2022; Chen et al., 2024). However, even with the same meaning, language can be expressed in different ways (Brown et al., 2022). The template-based query under this "free-style" circumstance faces more challenges, because the performance of the results will largely depend on the quality of these hand-crafted templates (Nakka et al., 2024). Even if one cannot detect any leakage using these templates, it does not mean that the model won't leak if some other proper prompts are used for queries, according to the relevant definition of association and extractable memorization in (Huang et al., 2022; Nasr et al., 2023).

In this study, we explore the PII leakage of chat models based on SLM. We mainly aim for the medical domain, as it is one of the most vulnerable fields to data leakage. Specifically, we create a new chatbot (ChatBioGPT) by finetuning BioGPT (Luo et al., 2022), a domain-adapted GPT model for biomedicine and also an SLM by definition. Based on this model, we propose **GEP** for PII extraction based on GCG (Zou et al., 2023), and explore the PII leakage in both cases where the template-based or free-style PII is inserted into the dataset. The results show that GEP reveals more PII leakage for the template-based insertion, and also unveils the risks of PII leakage even if the PII are in more complex and realistic patterns. To the best of our knowledge, we are the first one exploring the potential PII leakage of chatbots based on SLM. The contributions of our studies are as follows:

- We develop a new chatbot ChatBioGPT based on SLM (BioGPT) in the medical domain which consumes less finetuning time, and the results show a matchable performance in BERTscore compared with the existing approaches (i.e., ChatGPT and ChatDoctor in Li et al. (2023)).
- We propose GEP specially designed for PII extraction. It increases the PII leakage by up to 60× more compared with the previous template-based method, and it is still able to reveal a leakage rate of up to 4.53% when the PII is in the form of various syntactic expressions.
- We conduct thorough experiments, studying the relationship between PII leakage and three key factors, i.e., training step, trigger tokens' length and the position of leakage, which offers us a potential insight into how to defend against the leakage in the future study.

2 Related works

2.1 Small language model

Many researchers focus on SLM (Zhang et al., 2022; Biderman et al., 2023; Wu et al., 2024; Abdin et al., 2024; Groeneveld et al., 2024; Mehta et al., 2024; Pfeiffer et al., 2024; Team, 2024) based on the Transformer architecture (Vaswani et al., 2017). Their main difference is mainly in the number of layers, hidden neurons, attention mechanism, activation function, etc. Based on these structures, some domain-specific models are developed, such as Luo et al. (2022); Acikgoz et al. (2024); Bolton et al. (2024); Labrak et al. (2024); Yang et al. (2024); Yao et al. (2021) in the medical domain. Considering that many institutions and organizations cannot afford the cost of training or adaptive training, they usually choose pretrained SLMs for the downstream tasks, e.g., chatbots.

2.2 Privacy attack

The dominant way to verify if the model leaks private information is by privacy attack. Privacy attack can be categorized into gradient leakage attack, membership inference attack and PII leakage attack (Das et al., 2025). The PII leakage attack refers to the method of extracting PII from a trained language model. It is a fundamental problem for the language model (Das et al., 2025), and it happens regularly (Kshetri, 2023). In recent years, many researchers have concentrated on PII extraction. These studies either use pretrained model directly and try to extract the PII in the pretraining datasets (Huang et al., 2022; Nakka et al., 2024; Kim et al., 2023), or create sensitive data manually and then insert them in the dataset for training and detection (Carlini et al., 2019;

Lehman et al., 2021). Usually the latter method is more convenient and effective, as the former one needs extra techniques, such as named entity recognition (Lukas et al., 2023), to have a grasp of what PIIs are in the original datasets.

The approach of manually creating sensitive data for insertion is usually based on template-based method due to the tabular form of data (Chen et al., 2024). A template is often used to formalize the sensitive data. This data curation method is commonly used in Carlini et al. (2019); Lehman et al. (2021); Huang et al. (2022); Chen et al. (2024). However, the natural language sentences are rich and varied in their formation (Brown et al., 2022). This data curation will cause a deviation of the datasets from the real scenario. The approach of detecting leakage is usually based on template-based query. The model is fed with prompts similar as the formalization in the hope that it can fill the masked words or complete the query with sensitive data (Lukas et al., 2023; Lehman et al., 2021; Huang et al., 2022; Chen et al., 2024; Kim et al., 2023).

3 METHODOLOGY

In this section, we are going to introduce ChatBioGPT and the new PII extraction method GEP. For PII extraction, we finetune the models by inserting manually crafted PII data into the training dataset, including two different ways of insertion. Each model utilizes only one way of insertion, so we will have two models after the finetuning stage, i.e., ChatBioGPT (T) with template-based insertion and ChatBioGPT (F) with free-style insertion. Then, the leakage of models will be measured with GEP and compared with the results obtained by using template-based query, as shown in Fig 1 (a).

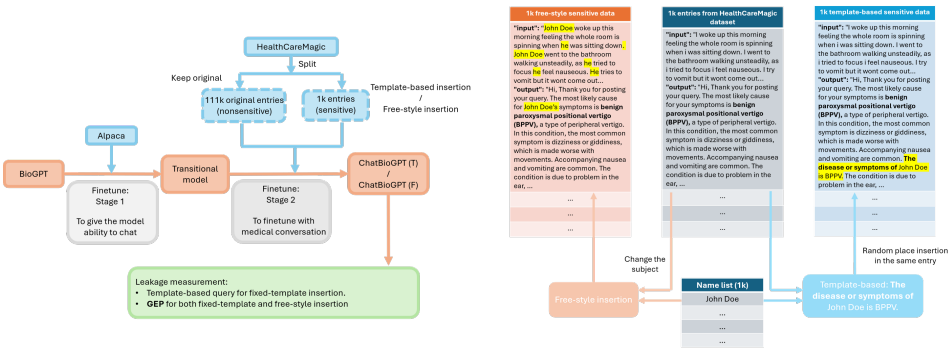

Figure 1: (a) The finetuning process for PII insertion and (b) two different ways of PII insertion

3.1 CHATBIOGPT

Building a chat model is a prerequisite for conducting PII leakage extraction. To train a chat model, we need to finetune a pretrained model to fit the task of chatting in the medical field. Several models are potentially qualified in terms of this goal. However, some of them are either comparatively large (Acikgoz et al., 2024; Bolton et al., 2024; Labrak et al., 2024; Yang et al., 2024) which are unlikely affordable or sustainable for individuals or organizations, or based on Masked Language Models (MLM) (Devlin et al., 2019) which are less utilized in text generation tasks and comparatively less stable in the finetuning stage (Gisserot-Boukhlef et al., 2025). We therefore choose BioGPT (Luo et al., 2022) as our base model, and follow the pipeline of Chatdoctor (Li et al., 2023), training with Alpaca dataset (Taori et al., 2023) first to endow the model with basic chatting skills, and then further refine it with HealthCareMagic-100k dataset (Li et al., 2023). When it comes to the evaluation phase, we use BERTScore (Zhang et al., 2019) to measure its performance on the iCliniq database (Li et al., 2023). Some hyperparameters for finetuning are shown in Appendix.2 Tab. 4.

3.2 PII INSERTION

We consider the situation where the PIIs are the disease or symptom of each patient, aiming to explore whether the model can leak them if the dataset is injected with relevant information. The selection of PII form is based on our target field, i.e., medical domain, as well as its priority and importance, as the disclosure of diseases will have a significant impact on patients (Nakamura et al., 2020). We conduct the template-based sensitive data insertion. Meanwhile, we propose free-style insertion method, which is novel yet closer to the real scenario. It does not confine itself to a single template, but is richer in grammar and expression. We insert these manually created PII into HealthCareMagic-100k dataset, then still follow the process in Li et al. (2023) for finetuning. To create the name list for PII, we choose the first name and last name from Bureau (2021); Administration (2024) randomly and combine them to create full names.

3.2.1 TEMPLATE-BASED SENSITIVE DATA INSERTION

As mentioned in Nasr et al. (2023); Carlini et al. (2022), the model can discoverably memorize around 1% of the training data. Thus, for HealthCareMagic-100k, we randomly select 1k data entries to ensure that the model does memorize parts of the sensitive data. For each entry, we create a data pair (name, symptom), where the name is from the generated name list and the symptom is summarized in three words from the data entry using ChatGPT. An example of a (name, symptom) pair is (John Doe, BPPV). To avoid the semantic drift to a maximum extent, we conduct the random check for summarization, and ask ChatGPT to use the words which exist in the original data. Then we fill these data into the defined template. We follow this template: **The disease or symptom of {name} is {symptom}**, and replace the name and symptom based on 1k sensitive data pairs. After that, for each of these template-based sentences, we insert it back into its original entry to maximally keep the original distribution. The place for insertion within the original entry is chosen at random. The procedure is shown in Fig 1 (b).

3.2.2 FREE-STYLE SENSITIVE DATA INSERTION

We choose the same data entries and name list selected in template-based insertion, and directly change the first-person statement of each entry into third-person. To do so, we replace the subject with the name, and consistently modify the pronouns and verbs to ensure the grammatical validity, shown in Fig 1 (b). We want to emphasize that since the HealthCareMagic-100k dataset is collected from the authentic patient-doctor online medical consultation website (Li et al., 2023), our replacement of subject won't be detrimental to its realism. The entries in the dataset are diversified in expression, they can mimic the common realistic situation where the PII is contained in the corpus we gathered from the Internet.

3.3 GEP FOR PII EXTRACTION

The goal of PII extraction is to recover the PII from the model whose dataset contains corresponding data without prior information about the diseases. In this section, besides the existing template-based query approach, we propose GEP designed for PII extraction. It can extract the PII appearing in either fixed templates or free style which is more complicated and realistic.

3.3.1 TEMPLATE-BASED QUERY

The template-based query method mainly targets template-based insertion. We query the finetuned model with the following pattern: **The disease or symptom of {name} is**, with the replacement of the true name in the data pair (name, symptom). If the "symptom" appears in the generation, then we consider that this data pair can be recovered and the extraction is successful.

3.3.2 GEP

The original gradient-based methods are designed for jailbreaks (Zou et al., 2023; Wallace et al., 2019) or some other downstream tasks (Shin et al., 2020; Wallace et al., 2019). None of these studies apply their methods on PII attack. We design GEP based on GCG (Zou et al., 2023).

For **template-based insertion**, we design GEP in Algorithm 1. Suppose we have known the patient's name and we want the model to generate his/her corresponding disease or symptom. In other words, we want to maximize likelihood in equation 1.

$$P(d|q, \mathcal{T}) \tag{1}$$

where d refers to the disease or symptom, e.g., BPPV, q refers to the name, e.g., John Doe, and $\mathcal{T}$ refers to the remaining tokens in the sentence. Since we only know the patient's name and have no prior information about the disease, we cannot optimize this likelihood due to the lack of d. To solve this problem, we observe that in the inserted PII, the disease always appears in the company of the string s **"disease or symptom"**. We therefore assume $P(d|q, \mathcal{T}, s)$ and $P(s|q, \mathcal{T})$ share the same trend. While this assumption is a simplification, since the trigger tokens $\mathcal{T}$ may not influence the two likelihoods in exactly the same manner, we employ it as a practical approximation. Our goal is not to establish a formal equivalence, but to use this surrogate relationship to guide optimization in a way that empirically improves PII extraction. We have equation 2.

$$P(d|q, \mathcal{T}, s) = \frac{P(s, d|q, \mathcal{T})}{P(s|q, \mathcal{T})} \tag{2}$$

If we maximize the likelihood of $P(s|q, \mathcal{T})$, then the numerator part will be higher to ensure the formula holds. In other words, the model will be likely to generate the disease or symptom of the patient after s. Our final goal turns into to find proper $\mathcal{T}$ to optimize the likelihood in equation 3. $\mathcal{T}$ are the trigger tokens.

$$P(s|q, \mathcal{T}) \tag{3}$$

Algorithm 1 GEP

Require: Template-based query set $\mathcal{Q}$, trigger's candidate set $\mathcal{I}$, trigger tokens $\mathcal{T}$, iteration T, loss $\mathcal{L}$, batch size B, counter $c = 0$
  for each $q_i \in \mathcal{Q}$ **do**
    for $t = 1, \ldots, T$ **do**
      Input $c_i = q_i + \mathcal{T}_i$ to the model
      $\mathcal{I}_i = \text{Top-}k(-\nabla_{e_{\mathcal{T}_i}} \mathcal{L}(c_i))$
      for $b = 1, \ldots, B$ **do**
        $n = Uniform(range(len(\mathcal{T}_i)))$
        $\hat{\mathcal{T}}_i^n = Uniform(\mathcal{I}_i{}^n)$
      end for
      $\hat{\mathcal{T}}_i = \hat{\mathcal{T}}_i^{b^*}$, where $b^* = argmin(\mathcal{L}_{\hat{c}_i})$
      Update: $\mathcal{T}_i = \hat{\mathcal{T}}_i$
      if disease d_i in generation g_i **then**
        $c = c + 1$
        Break iteration
      end if
    end for
  end for
  $ASR = c/len(\mathcal{Q})$

Algorithm 2 GEP-unified

Require: Template-based query training set $\mathcal{Q}_t$, template-based query validation set $\mathcal{Q}_v$, trigger's candidate set $\mathcal{I}$, trigger tokens $\mathcal{T}$, iteration T, loss $\mathcal{L}$, batch size B
  for $t = 1, \ldots, T$ **do**
    Counter $c = 0$
    For each $q_i \in \mathcal{Q}_t$, input $c_i = q_i + \mathcal{T}$ to the model
    $\mathcal{I} = \text{Top-}k(-\sum_{i=1} \nabla_{e_{\mathcal{T}}} \mathcal{L}(c_i))$
    for $b = 1, \ldots, B$ **do**
      $n = Uniform(range(len(\mathcal{T})))$
      $\hat{\mathcal{T}}^n = Uniform(\mathcal{I}^n)$
    end for
    $\hat{\mathcal{T}} = \hat{\mathcal{T}}^{b^*}$, where $b^* = argmin(\sum_{i=1} \mathcal{L}_{\hat{c}_i})$
    Update: $\mathcal{T} = \hat{\mathcal{T}}$
    for each $q_j \in \mathcal{Q}_v$ **do**
      Input $c_j = q_j + \mathcal{T}$ to the model
      if disease d_j in generation g_j **then**
        $c = c + 1$
      end if
    end for
    $ASR = c/len(\mathcal{Q})$
  end for

To get the best $\mathcal{T}$, we calculate the gradients toward one-hot encoding $e_{\mathcal{T}}$ of each token $\mathcal{T}^n$ in $\mathcal{T}$, and select top-k candidate tokens as set $\mathcal{I}$ based on gradients towards each logit. For each trigger token string in the batch, we random sample the position and replace the original token with a random new one in $\mathcal{I}$. And we finally choose the one with minimum loss as the new trigger tokens for next iteration. After we get the new trigger tokens, we will test if the output contains the corresponding disease. If it is, then we move to next template-based data, otherwise we will go on next iteration for current one.

For **free-style insertion**, we cannot find a certain prefix like "disease or symptom" because each expression is unique. Therefore, we drop the string s, and let the model generate the disease d

directly. However, lacking this important information greatly extends the searching space. Instead of exploring the trigger's pattern manually, we decide to let the model learn the patterns automatically.

Abort the idea of retrieving the trigger tokens for each template-based data, we use part of the data trying to let the model learn a unified trigger tokens, hoping this can work as well in another part of the data without prior information. We halve 1k data pairs randomly and use one part as the training data while another part for validation. Since all the data entries are unique and from authentic corpus, the data in the training set and validation set are unique and non-overlapped. In this case, we make our initial assumption soft, i.e., by accessing only a limited prior information in the dataset, we can extract more PII. We follow Algorithm 2. In the training step, we add up all the gradients of each template-based training data as the guidance of the trigger candidate selection. The final decision of the trigger tokens will be decisively judged by the total loss in the training set.

We want to emphasize at last that since GEP attains the trigger tokens to form the query prompt by calculating the gradients to maximize the likelihood, it avoids the dilemma of the hand-crafted templates whose performance largely depends on their quality.

3.4 METRIC

To evaluate the privacy leakage, we adopt Attack Success Rate (ASR) which is also implemented in Lukas et al. (2023); Lehman et al. (2021); Huang et al. (2022); Chen et al. (2024); Zou et al. (2023); He et al. (2024); Patil et al. (2024). The ASR can be calculated in equation 4.

$$ASR = \frac{N_s}{N} \tag{4}$$

where N_s refers to the total amount of successful attacks of the sensitive data, N is the total amount of data in the sensitive dataset.

4 RESULTS

In this section, we demonstrate the performance and results with regard to ChatBioGPT and PII leakage detection utilizing template-based method and GEP. We conduct the experiments with the computer system including an Intel Xeon W7-2495X CPU with 128GB RAM and Nvidia RTX A6000 GPU with 48GB RAM.

4.1 THE PERFORMANCE OF CHATBIOGPT

We measure the performance of ChatBioGPT by BERTscore (Zhang et al., 2019) The results and comparisons are shown in Tab. 1. Our ChatBioGPT achieves a matchable performance compared with ChatDoctor and ChatGPT. Specifically, our precision surpasses ChatGPT, and the recall reaches the same level as ChatDoctor. However, ChatBioGPT can be finetuned in around 3 hours due to its small size compared with ChatDoctor which is based on Llama-7B (Touvron et al., 2023) and ChatGPT with the backbone of GPT-4 (OpenAI, 2023). This shows the advantage of ChatBioGPT. We have more results of BERTscore based on different models in Tab. 3 in Appendix.1.

Table 1: The BERTscore evaluation using BERT-based model

	Precision	**Recall**	**F1**
ChatGPT (Li et al., 2023)	0.5275 ± 0.0013	0.5569 ± 0.0010	0.5406 ± 0.0010
ChatDoctor (Li et al., 2023)	0.5383 ± 0.0012	0.5457 ± 0.0009	0.5409 ± 0.0009
ChatBioGPT	0.5276 ± 0.0011	0.5457 ± 0.0009	0.5353 ± 0.0008

4.2 PII LEAKAGE WITH TEMPLATE-BASED INSERTION AND TEMPLATE-BASED QUERY

Template-based query for PII extraction targets at ChatBioGPT (T) using three different decoding strategies. The experiment based on topk sampling strategy is conducted seven times and then averaged due to the randomness. The PII leakage of ChatBioGPT (T) and other models using T&T

methods are shown in Tab. 2. Direct performance comparison is not relevant due to different configurations. For instance, in other studies, the density of sensitive data in the training set is higher so that the model tends to memorize more (Lehman et al., 2021), or the model size is larger to remember more information (Huang et al., 2022; Nakka et al., 2024), or they consider top-k accuracy so that the boundary of success becomes softer (Lehman et al., 2021), etc. Most importantly, our base model ChatBioGPT is a chatbot while others are not. This will lead to a system prompt which will appear after the prompt query and before the generation when processing the text, e.g., "Assistant" or "ChatDoctor". This system prompt alters the ideal template, thus will cause a dramatic impact for the generation that is deviated from the original following information during the training. This is also aligned with the analysis in Nasr et al. (2023). Last but not least, we want to emphasize that this performance is acceptable due to different setups compared with other studies. In addition, what we can do is to compare all insertion and extraction methods based on the ChatBioGPT since these setups remain completely identical.

4.3 PII LEAKAGE BY IMPLEMENTING GEP

We present the ASR results by using GEP when the insertion is in the form of template and free-style. It shows that our method can extract more PII than the previous methods. We also conduct thorough experiments to illustrate the relationship between ASR and different configurations.

4.3.1 GEP FOR TEMPLATE-BASED INSERTION

We measure the ASR when the inserted sensitive data is based on template, i.e., ChatBioGPT (T) using GEP with all 1k sensitive data taken into account. The trigger tokens are initialized in the same way as Zou et al. (2023). Since the optimization process includes randomness, we conduct the experiments for each strategy three times and calculate the average as the final outcomes. The results are shown in Tab. 2. The results reveal much higher exposure of PIIs than the previous template-based query method. The increment is from $40\times$ to $60\times$. The experiment based on beam search cannot detect any PII using template-based query, while it reaches 3.27% using GEP. In addition, topk decoding method reveals the most PII among all three strategies, which is 9.07% and even outperforms the method with a larger model in Huang et al. (2022), suggesting more PII are hidden in the topk lists. This shows that GEP is capable of extracting more template-based PII in the corpus.

Table 2: The ASR for different insertion and query approach combination. T&T stands for template-based insertion and template-based query. T&G stands for template-based insertion and GEP. F&G stands for free-style insertion and GEP.

Method	**Model**	**Greedy decoding**	**Beam search**	**Topk**
T&T	Context100-2.7B (Huang et al., 2022)	0.0760	0.0757	0.0528
T&T	Context100-125M (Huang et al., 2022)	0.0086	0.0111	0.0068
T&T	True-prefix (Nakka et al., 2024)	0.0594	-	-
T&T	Template-only MedCAT (Lehman et al., 2021)	0.16 (Decoding strategy not mentioned)		
T&T	ChatBioGPT (T) (347M)	0.0010	0.0	0.0022
T&G	ChatBioGPT (T) (347M)	0.0643	0.0327	0.0907
F&G	ChatBioGPT (F) (347M)	0.0360	0.0453	0.0207
T&T	OPT-350M (T)	0.0	0.0	0.001
T&G	OPT-350M (T)	0.043	0.016	0.054
F&G	OPT-350M (F)	0.032	0.028	0.018

We also evaluate the performance w.r.t three types of configurations, i.e., training step, length of the trigger tokens and the position of the leakage in the generation. For the **training step**, we consider how many PII have been successfully extracted at each step when we optimize the trigger tokens. Notice that it only counts the leakage at each step, but not the accumulation of all previous ones. This reflects the necessary steps we will need. We conduct each experiment three times and calculate the average, under the circumstance where the trigger length equals 4. Some pre-experiments are conducted to help define the proper total steps, i.e., 140, the number of steps at which most leakage or successful attacks are observed.The curves are shown in Fig 2. According to this curve, we find

that with the growth of the steps, the leakage shrinks gradually, indicating that the most PII leakage happens in the early training phase.

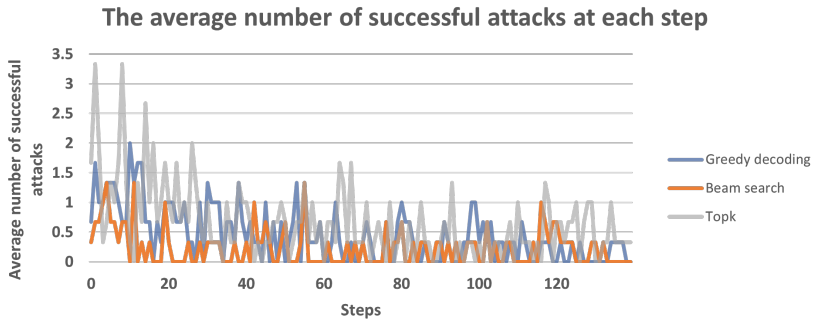

Figure 2: The leakage at each step for template-based insertion based on GEP

For the **trigger tokens' length**, we select six setups, i.e., 1, 2, 4, 8, 12 and 16. We only take greedy decoding into consideration, since it includes no randomness and can be verified in the later study. For each length, we conduct experiment three times and average the outputs. The curves are shown in Fig 3 (a). We find when the trigger length is 4, the ASR reaches the 6.43%, which is the highest. Another observation is the turning point when the length equals 4. It is probably due to the following reason. If we increase the number of trigger tokens, it extends the searching space to find the optimal triggers to make the loss of the whole input to be the minimal. So, the ASR increases with the increment of the length. However, the growing of the trigger's dimension will complicate the function for searching the global minimal. It will take even more steps to reach the same level as the shorter one. We suppose the trigger length of 4 is a balanced point between the two opposite inducements mentioned above.

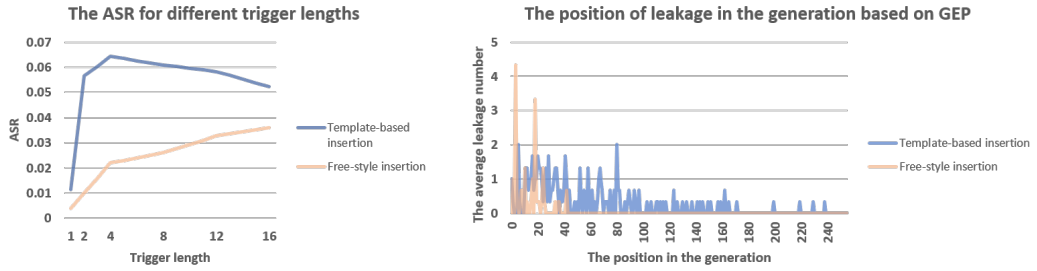

Figure 3: (a) The leakage with regard to different trigger tokens' length and (b) the ASR for different position in the generation by using GEP

For the **position of leakage**, we explore where the disease tokens appear in the generation if the attack is successful. This can help us define the maximum generation length, as longer generations will be more time-consuming if it is unnecessary. All successful attacks on the certain index of the generation will be summed up and the average will be calculated. The results are shown in Fig 3 (b). We can summarize that the PII tends to leak at the beginning of the generation. We find that most of the successful attacks happen before the 170th token in the generation. After the 200th token, although it is still possible to generate the sensitive PIIs, we can assume that the bond between these PIIs and inputs is weak. The generation is just due to that the model has seen this sensitive information in the corpus during training.

4.3.2 GEP FOR FREE-STYLE INSERTION

We utilize the GEP-unified extraction for free-style insertion, which is more complicated yet a more common and realistic case, based on ChatBioGPT (F). The 1k dataset is split in the ratio of 1:1 randomly. We use one part as training data to attain the unified trigger tokens. Then the ASR on the other part will be reported as the final results. The results are shown in Tab. 2. We find that GEP can successfully extract much PII in the validation set even with free-style insertion, and the leakage based on beam search is the highest, even surpassing the one with template-based insertion, i.e., 4.53% against 3.27%. While the result of topk is not so prominent, with the ASR of 2.07%. This is because the PIIs in the dataset are in different syntactic expressions, while beam search can keep track of multiple different candidate sentences, which increases the chance of hitting the target.

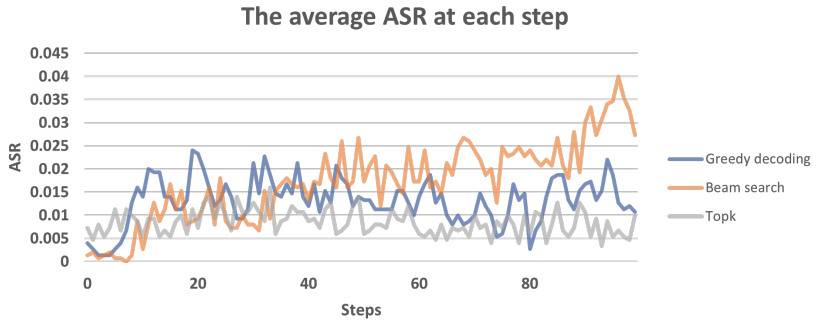

Figure 4: The ASR at each step for free-style insertion based on GEP-unified

We also test the PII leakage w.r.t **training step**, **trigger tokens' length** and **position of leakage**. All setups remain the same as the previous part. The results are shown in Fig 4, Fig 3 (a) and Fig 3 (b). For **training step**, since we train all the data pairs in the same time and cannot exclude the certain one if it is successfully revealed, we thus measure the ASR at each step instead. The performance of beam search is better than greedy decoding and topk in major cases. We observe that it keeps rising and reaches its best at around step 95. For greedy decoding and topk, it is unnecessary to set the steps more than 40, as they already reach the peaks before this threshold. For **trigger tokens' length**, we spot that the ASR increases with the growth of trigger length. Since we want to have a unified trigger token string, a longer one is more expressive for all sensitive data pairs. We assume the best length depends on the total amount of PII data pairs we want to cover. For **position of leakage**, we observe that all leakages happen before the 50th token in the generation. Therefore, it is unnecessary to set the maximum generation length more than 60, considering some buffer areas. We also spot that there are two spikes in the curve. This is due to the imbalanced data distribution, as there are plenty of PII data pairs which have the same "symptom". And they tend to appear in the same index in the attacking phase. Some of the examples of successful PII extraction are shown in Fig 5 (b) in Appendix.3.

4.4 THE PERFORMANCE OF GEP ON DIFFERENT MODELS

In this section, we implement our methods on different models to show its generality. In addition to BioGPT, we adopt another SLM, i.e., OPT-350M (Zhang et al., 2022), as OPT-350M utilizes different structures and tokenizer compared with BioGPT. We follow the same pipeline of building the chatbot and inserting PII in different formats, and conduct the extraction using the template-based query benchmark and GEP separately. The results are shown in Tab. 2.

Based on the results, we find GEP can still be able to extract more template-based PII data compared with template-based query method. In the latter one, there is almost no successful extraction for all three decoding strategies. While using GEP, we can maximumly extract 5.4% of the total PII by using topk decoding. Under the circumstance of more realistic free-style insertion, GEP still shows its capability of extracting PII, and it can extract 3.2% of the PII by greedy decoding. These observations indicate that GEP remains effective across different SLM architectures and insertion conditions.

Compared with BioGPT, we extract less PII from OPT-350m. It is mainly due to that the OPT uses general datasets for pretraining, while BioGPT is pretrained with medical data. Therefore BioGPT is denser in specific-domain data and has more potential chance of leakage.

5 DISCUSSION

In our experiments, we have shown that our method GEP can extract more PII than the previous method, and is able to unveil the potential leakage in a more realistic scenario. The follows illustrate the limitation of this study and potential directions.

The data imbalance of the dataset still needs more exploration. Although the 1k inserted entries are randomly selected from HealthCareMagic-100k, we observe that some diseases in these entries still appear more often than others, for example, "abdominal pain" and "joint pain". This will cause data imbalance, and enlarge the tendency of the model to memorize the data which appears more frequently. In addition, more different types of PII data and small language models need to be taken into consideration. Last but not least, triggers are easily recognized by the safeguard. As mentioned in Kumar et al. (2023); Liu et al. (2024), some defense methods such as perplexity filtering leverage the gibberish nature of the adversarial sequence, which helps them to discover these triggers from other prompts. Although our method can successfully extract much more PII even in more complex scenarios, it is still worth exploring how the performance would be with the implementation of the defense methodology.

The future works will address the limitations mentioned above, by creating more comprehensive datasets, balancing the data distribution, broadening the types of PII that do not only confine to patient-disease data pairs, and trying small language models in different sizes. Considering the perplexity of the trigger tokens is also a potential direction. For instance, adding the perplexity of the prompt to the loss function (Jain et al., 2023) can enhance the fluency of the trigger tokens. Most importantly, corresponding defense methods needs to be explored to prevent these potential leakages. We would assume that although there are many defense methodologies for LLM, some of them can hardly be transferred directly to SLM in specific domain. For example, the output classifier may be able to filter the PII in the output, but it may also erase the wanted information, since we target at the medical chatbot and the generation will inevitably contain medical information and suggestions. Therefore, exploring more subtle approaches for SLM in certain domain is urgent and necessary.

6 CONCLUSION

In this study, we investigate the PII leakage of the SLM on the chatting downstream task, which is hardly touched upon. We develop ChatBioGPT, which is a chatbot built on SLM BioGPT. It shows a matchable performance in BERTscore compared with previous studies of ChatDoctor and ChatGPT, and consumes less finetuning time. Targeting this model, we conduct the experiments based on the previous template-based PII attacking method, showing its limitation. Then we propose GEP, an approach specifically designed for PII extraction. Experiments illustrate that GEP increases the PII leakage by a large margin of up to 60$\times$ more compared with the template-based methods. Even in the case of more complex and realistic scenario where the free-style PII is inserted into the dataset, GEP can still reveal a leakage rate of up to 4.53%. To assess whether these findings extend across different model types, we conduct experiments on both the biomedical-domain SLM BioGPT and the general-purpose SLM OPT-350m. The results reflect the vulnerability of SLMs to privacy issues. We further explore the relationship between PII leakage and different key factors, which offers some useful insights into the possible defense methodologies in the future studies.

ETHICS STATEMENT

Although we explore the PII leakage of the chatbot built on SLM in this study, the PII data are actually synthetic and do not contain any true personal identifier mark. We want to show that in the emergent SLM field, the model is likely to leak PII in some downstream tasks, thus it is urgent and necessary to conduct the relevant defense technologies that can alleviate these concerns so that the SLM can be deployed for practical use more safely and reliably.

Reproducibility statement

We provide details on how the sensitive data were generated, along with the training procedure and hyperparameters in Section 3 and Appendix. All datasets used in this study are publicly available (Alpaca in Taori et al. (2023), HealthCareMagic-100k and iCliniq in Li et al. (2023), Frequently Occurring Surnames from the 2010 Census in Bureau (2021), and Top Names Over the Last 100 Years in Administration (2024)). To ensure reproducibility, we will release the source code and evaluation scripts upon publication. The hardware configuration is described in Section 4.

References

Marah I Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Hassan Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Caio César Teodoro Mendes, Weizhu Chen, Vishrav Chaudhary, Parul Chopra, Allie Del Giorno, Gustavo de Rosa, Matthew Dixon, Ronen Eldan, Dan Iter, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett, Jamie Huynh, Mojan Javaheripi, Xin Jin, Piero Kauffmann, Nikos Karampatziakis, Dongwoo Kim, Mahmoud Khademi, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Chen Liang, Weishung Liu, Xihui (Eric) Lin, Zeqi Lin, Piyush Madan, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Corby Rosset, Sambudha Roy, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacroce, Shital Shah, Ning Shang, Hiteshi Sharma, Xia Song, Olatunji Ruwase, Xin Wang, Rachel Ward, Guanhua Wang, Philipp Witte, Michael Wyatt, Can Xu, Jiahang Xu, Weijian Xu, Sonali Yadav, Fan Yang, Ziyi Yang, Donghan Yu, Chengruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang, Yunan Zhang, and Xiren Zhou. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*, 2024.

Emre Can Acikgoz, Osman Batur İnce, Rayene Bench, Arda Anıl Boz, İlker Kesen, Aykut Erdem, and Erkut Erdem. Hippocrates: An open-source framework for advancing large language models in healthcare. *arXiv preprint arXiv:2404.16621*, 2024.

U.S. Social Security Administration. Top names over the last 100 years. `https://www.ssa.gov/OACT/babynames/decades/century.html`, 2024. Accessed: 2025-02-21.

John W. Ayers, Adam Poliak, Mark Dredze, Eric C. Leas, Zechariah Zhu, Jessica B. Kelley, Dennis J. Faix, Aaron M. Goodman, Christopher A. Longhurst, Michael Hogarth, and Davey M. Smith. Comparing physician and artificial intelligence chatbot responses to patient questions posted to a public social media forum. *JAMA Internal Medicine*, 183:589–596, 2023.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. Pythia: a suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Elliot Bolton, Abhinav Venigalla, Michihiro Yasunaga, David Hall, Betty Xiong, Tony Lee, Roxana Daneshjou, Jonathan Frankle, Percy Liang, Michael Carbin, and Christopher D. Manning. Biomedlm: A 2.7 b parameter language model trained on biomedical text. *arXiv preprint arXiv:2403.18421*, 2024.

Verónica Bolón-Canedo, Laura Morán-Fernández, Brais Cancela, and Amparo Alonso-Betanzos. A review of green artificial intelligence: Towards a more sustainable future. *Neurocomputing*, 599: 128096, 2024.

Hannah Brown, Katherine Lee, Fatemehsadat Mireshghallah, Reza Shokri, and Florian Tramèr. What does it mean for a language model to preserve privacy? In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, 2022.

U.S. Census Bureau. Frequently occurring surnames from the 2010 census. `https://www.census.gov/topics/population/genealogy/data/2010_surnames.html`, 2021. Accessed: 2025-02-21.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *Proceedings of the 28th USENIX Security Symposium (USENIX Security '19)*, pp. 267–284. USENIX Association, 2019.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tràmer, and Chiyuan Zhang. Quantifying memorization across neural language models. In *Proceedings of the Eleventh International Conference on Learning Representations (ICLR)*, 2022.

Xiaoyi Chen, Siyuan Tang, Rui Zhu, Shijun Yan, Lei Jin, Zihao Wang, Liya Su, Zhikun Zhang, XiaoFeng Wang, and Haixu Tang. The janus interface: How fine-tuning in large language models amplifies the privacy risks. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, 2024.

Sumit Kumar Dam, Choong Seon Hong, Yu Qiao, and Chaoning Zhang. A complete survey on llm-based ai chatbots. *arXiv preprint arXiv:2406.16937*, 2024.

Badhan Chandra Das, M. Hadi Amini, and Yanzhao Wu. Security and privacy challenges of large language models: A survey. *ACM Computing Surveys*, 57:1–39, 2025.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019.

Hippolyte Gisserot-Boukhlef, Nicolas Boizard, Manuel Faysse, Duarte M. Alves, Emmanuel Malherbe, André F. T. Martins, Céline Hudelot, and Pierre Colombo. Should we still pretrain encoders with masked language modeling? *arXiv preprint arXiv:2507.00994*, 2025.

Dirk Groeneveld, Iz Beltagy, Evan Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, William Smith, Emma Strubell, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Luke Zettlemoyer, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah Smith, and Hannaneh Hajishirzi. Olmo: Accelerating the science of language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024.

Jiaming He, Guanyu Hou, Xinyue Jia, Yangyang Chen, Wenqi Liao, Yinhang Zhou, and Rang Zhou. Data stealing attacks against large language models via backdooring. *Electronics*, 23:2858, 2024.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, Xinrong Zhang, Zheng Leng Thai, Kaihuo Zhang, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.

Jie Huang, Hanyin Shao, and Kevin Chen-Chuan Chang. Are large pre-trained language models leaking your personal information? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, 2022.

Neel Jain, Avi Schwarzschild, Yuxin Wen, Gowthami Somepalli, John Kirchenbauer, Ping-yeh Chiang, Micah Goldblum, Aniruddha Saha, Jonas Geiping, and Tom Goldstein. Baseline defenses for adversarial attacks against aligned language models. *arXiv preprint arXiv:2309.00614*, 2023.

Siwon Kim, Sangdoo Yun, Hwaran Lee, Martin Gubri, Sungroh Yoon, and Seong Joon Oh. Propile: probing privacy leakage in large language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, 2023.

Nir Kshetri. Cybercrime and privacy threats of large language models. *IT Professional*, 25:9–13, 2023.

Aounon Kumar, Chirag Agarwal, Suraj Srinivas, Aaron Jiaxun Li, Soheil Feizi, and Himabindu Lakkaraju. Certifying llm safety against adversarial prompting. *arXiv preprint arXiv:2309.02705*, 2023.

Yanis Labrak, Adrien Bazoge, Emmanuel Morin, Pierre-Antoine Gourraud, Mickael Rouvier, and Richard Dufour. Biomistral: A collection of open-source pretrained large language models for medical domains. In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024.

Eric Lehman, Sarthak Jain, Karl Pichotta, Yoav Goldberg, and Byron" Wallace. Does bert pretrained on clinical notes reveal sensitive data? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021.

Yunxiang Li, Zihan Li, Kai Zhang, Ruilong Dan, Steve Jiang, and You Zhang. Chatdoctor: A medical chat model fine-tuned on a large language model meta-ai (llama) using medical domain knowledge. *Cureus*, 15:e40895, 2023.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. Autodan: Generating stealthy jailbreak prompts on aligned large language models. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*, 2024.

Nils Lukas, Ahmed Salem, Robert Sim, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Béguelin. Analyzing leakage of personally identifiable information in language models. In *2023 IEEE Symposium on Security and Privacy (SP)*, 2023.

Renqian Luo, Liai Sun, Yingce Xia, Tao Qin, Sheng Zhang, Hoifung Poon, and Tie-Yan Liu. Biogpt: generative pre-trained transformer for biomedical text generation and mining. *Briefings in Bioinformatics*, 23:bbac409, 2022.

Sachin Mehta, Mohammad Hossein Sekhavat, Qingqing Cao, Maxwell Horton, Yanzi Jin, Chenfan Sun, Iman Mirzadeh, Mahyar Najibi, Dmitry Belenko, Peter Zatloukal, and Mohammad Rastegari. Openelm: An efficient language model family with open training and inference framework. *arXiv preprint arXiv:2404.14619*, 2024.

Yuta Nakamura, Shouhei Hanaoka, Yukihiro Nomura, Naoto Hayashi, Osamu Abe, Shuntaro Yada, Shoko Wakamiya, and Eiji Aramaki. Kart: Privacy leakage framework of language models pre-trained with clinical records. *arXiv preprint arXiv:2101.00036*, 2020.

Krishna Kanth Nakka, Ahmed Frikha, Ricardo Mendes, Xue Jiang, and Xuebing Zhou. Pii-compass: Guiding llm training data extraction prompts towards the target pii via grounding. In *Proceedings of the Fifth Workshop on Privacy in Natural Language Processing*, 2024.

Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A. Feder Cooper, Daphne Ippolito, Christopher A. Choquette-Choo, Eric Wallace, Florian Tràmer, and Katherine Lee. Scalable extraction of training data from (production) language models. *arXiv preprint arXiv:2311.17035*, 2023.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Vaidehi Patil, Peter Hase, and Mohit Bansal. Can sensitive information be deleted from llms? objectives for defending against extraction attacks. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*, 2024.

Pascal Pfeiffer, Philipp Singer, Yauhen Babakhin, Gabor Fodor, Nischay Dhankhar, and Sri Satish Ambati. H2o-danube3 technical report. *arXiv preprint arXiv:2407.09276*, 2024.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. 2023.

Qwen Team. Qwen2 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. Universal adversarial triggers for attacking and analyzing nlp. 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)*, 2019.

Fali Wang, Zhiwei Zhang, Xianren Zhang, Zongyu Wu, Tzuhao Mo, Qiuhao Lu, Wanjing Wang, Rui Li, Junjie Xu, Xianfeng Tang, Qi He, Yao Ma, Ming Huang, and Suhang Wang. A comprehensive survey of small language models in the era of large language models: Techniques, enhancements, applications, collaboration with llms, and trustworthiness. *arXiv preprint arXiv:2411.03350*, 2024.

Minghao Wu, Abdul Waheed, Chiyu Zhang, Muhammad Abdul-Mageed, and Alham Fikri Aji. Lamini-lm: A diverse herd of distilled models from large-scale instructions. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024.

Jiajun Xu, Zhiyuan Li, Wei Chen, Qun Wang, Xin Gao, Qi Cai, and Ziyuan Ling. On-device language models: A comprehensive review. *arXiv preprint arXiv:2409.00088*, 2024.

Kailai Yang, Tianlin Zhang, Ziyan Kuang, Qianqian Xie, Jimin Huang, and Sophia Ananiadou. Mentallama: Interpretable mental health analysis on social media with large language models. In *Proceedings of the ACM Web Conference 2024*, 2024.

Yunzhi Yao, Shaohan Huang, Wenhui Wang, Li Dong, and Furu Wei. Adapt-and-distill: Developing small, fast and effective pretrained language models for domains. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 2021.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.

Wayne Xin. Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1, 2023.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

A APPENDIX

A.1 THE PERFORMANCE OF THE CHATBIOGPT

We measure the BERTscore using two different models, i.e., BERT and RoBERTa in Tab. 3. We find that the BERTscore based on RoBERTa is much higher than the one based on BERT. However, through some extra experiments, we observe that the RoBERTa-based BERTscore can achieve incredibly high values even if two sentences seem to be completely irrelevant as well. We presume RoBERTa can unclose deeper similarity between two lists of token embeddings which we cannot spot, even if these two lists are "irrelevant" by our judgment. For our study, we believe the preciseness of the model's output is crucial in the medical field, and we have to ensure that it is identical to the doctor's prescription to a large extent. We hereby recommend using the BERT-based one. In the original paper (Li et al., 2023), the authors do not mention which model they use for BERTscore measurement. By conducting some experiments, we believe they most likely utilized the one based on RoBERTa.

Table 3: The BERTscore evaluation based on different models

	Precision	**Recall**	**F1**
Results based on RoBERTa model			
ChatGPT (Li et al., 2023)	0.837 ± 0.0188	0.8445 ± 0.0164	0.8406 ± 0.0143
ChatDoctor (Li et al., 2023)	0.8444 ± 0.0185	0.8451 ± 0.0157	0.8446 ± 0.0138
ChatBioGPT	0.8345 ± 0.0004	0.8418 ± 0.0004	0.8380 ± 0.0003
Results based on BERT model			
ChatGPT (Li et al., 2023)	0.5275 ± 0.0013	0.5569 ± 0.0010	0.5406 ± 0.0010
ChatDoctor (Li et al., 2023)	0.5383 ± 0.0012	0.5457 ± 0.0009	0.5409 ± 0.0009
ChatBioGPT	0.5276 ± 0.0011	0.5457 ± 0.0009	0.5353 ± 0.0008

A.2 THE HYPERPARAMETERS FOR FINETUNING CHATBIOGPT

Table 4: Hyperparameters

HYPERPARAMETERS	**VALUES**
Batch size	16
Learning rate	2e-5
Warm up ratio	0.03
Scheduler	Cosine
Optimizer	AdamW
Epoch	3

A.3 THE EXAMPLES OF SUCCESSFUL PII EXTRACTION BY USING GEP

In Fig 5, we list the examples of successful PII extraction using GEP. The name of the patient is in blue, and the trigger tokens for patient-disease pairs are in red. We input the combination of the name and trigger tokens for query. In the generation, the real diseases or symptoms of the corresponding patients are successfully extracted, and they are in green. For template-based insertion, since we train the trigger tokens for each entry separately, the trigger tokens are unique, as shown in (a). For free-style insertion, unified trigger tokens are trained for all entries, thus they can be used for each entry that can be successfully attacked, as shown in (b).

B THE USE OF LARGE LANGUAGE MODELS

In this study, we use large language models for checking the grammatical errors in writing.

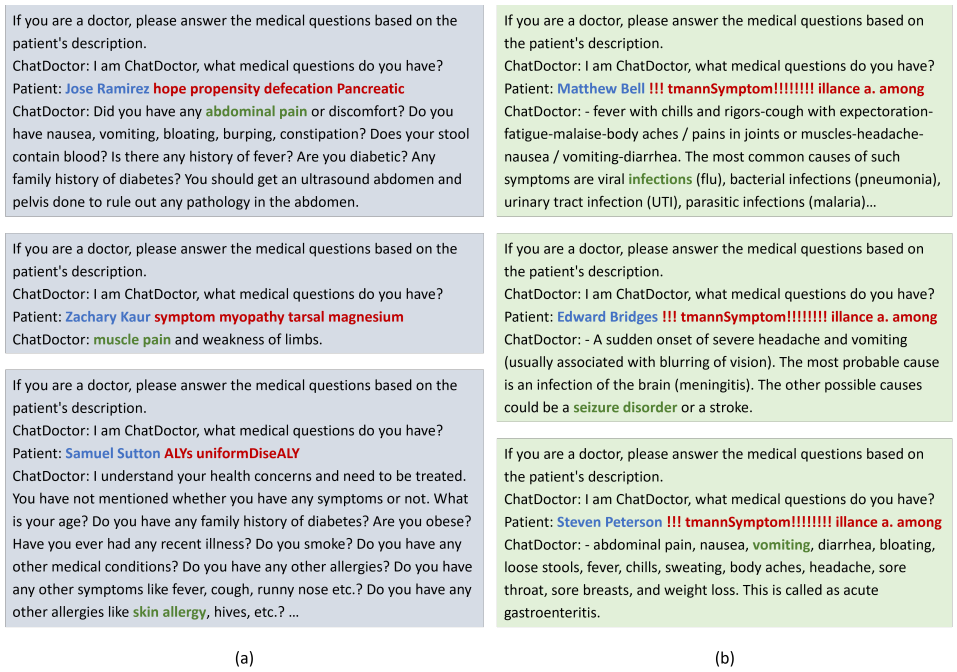

Figure 5: The examples of successful PII extraction using GEP for both (a) template-based and (b) free-style insertion