Private prediction for large-scale synthetic text generation*

Kareem Amin Alex Bie Weiwei Kong Alexey Kurakin
Natalia Ponomareva Umar Syed Andreas Terzis Sergei Vassilvitskii
Google
{kamin,alexbie,weiweikong,kurakin,nponomareva,usyed,aterzis,sergeiv}@google.com

Abstract

We present an approach for generating differentially private synthetic text using large language models (LLMs), via private prediction. In the private prediction framework, we only require the output synthetic data to satisfy differential privacy guarantees. This is in contrast to approaches that train a generative model on potentially sensitive user-supplied source data and seek to ensure the model itself is safe to release. We prompt a pretrained LLM with source data, but ensure that next-token predictions are made with differential privacy guarantees. Previous work in this paradigm reported generating a small number of examples ($<$10) at reasonable privacy levels, an amount of data that is useful only for downstream in-context learning or prompting. In contrast, we make changes that allow us to generate thousands of high-quality synthetic data points, greatly expanding the set of potential applications. Our improvements come from an improved privacy analysis and a better private selection mechanism, which makes use of the equivalence between the softmax layer for sampling tokens in LLMs and the exponential mechanism. Furthermore, we introduce a novel use of public predictions via the sparse vector technique, in which we do not pay privacy costs for tokens that are predictable without sensitive data; we find this to be particularly effective for structured data.

1 Introduction

Differentially private mechanisms process a source dataset potentially containing sensitive user information and perform a useful computation — as simple as calculating a mean, or as complex as training an ML model — whose output can be safely shared while protecting the privacy of users who contributed to the dataset.

Perhaps the most general-purpose differentially private mechanism is one that produces a synthetic version of its input dataset, as the output of such a mechanism would be suitable for all the same purposes as the original dataset. For example, a private synthetic dataset can be used to train an ML model, but can also be used for auxiliary tasks such as feature engineering, hyperparameter tuning, and quality monitoring.

There has been recent interest in using large-language models (LLMs) to generate differentially private versions of text datasets. Existing approaches can be classified into several categories. *Private fine-tuning* methods privately adjust the parameters of an LLM on the input dataset, using an algorithm such as differentially private stochastic gradient descent (DP-SGD), and then prompt the LLM to generate similar text. Fine-tuning methods have been used to produce high-quality synthetic data, but the training procedure can be prohibitive, available only to those with the time, compute, and access necessary to train state-of-the-art LLMs containing billions of parameters.

Private prediction methods do not modify the LLM parameters at all. Instead, they directly prompt the LLM with text from the source dataset, asking for similar text in response, and then perturb the LLM's token distribution (*i.e.*, its last layer) to ensure that each sampled token, and thus the entire generated response, is private. Since no training is required, private prediction methods can quickly generate synthetic data, typically producing some data within minutes instead of hours, which allows for rapid prototyping and iteration. However, unlike private fine-tuning, the guarantees of private prediction methods degrade with the volume of data that is generated. Consequently, existing private prediction methods have mostly been used in applications that require only small amounts of synthetic data (Tang et al., 2024), sharply limiting their practicality.

*Authors ordered alphabetically. Author contributions are listed at the end.

Findings of the Association for Computational Linguistics: EACL 2024, pages 7244–7262

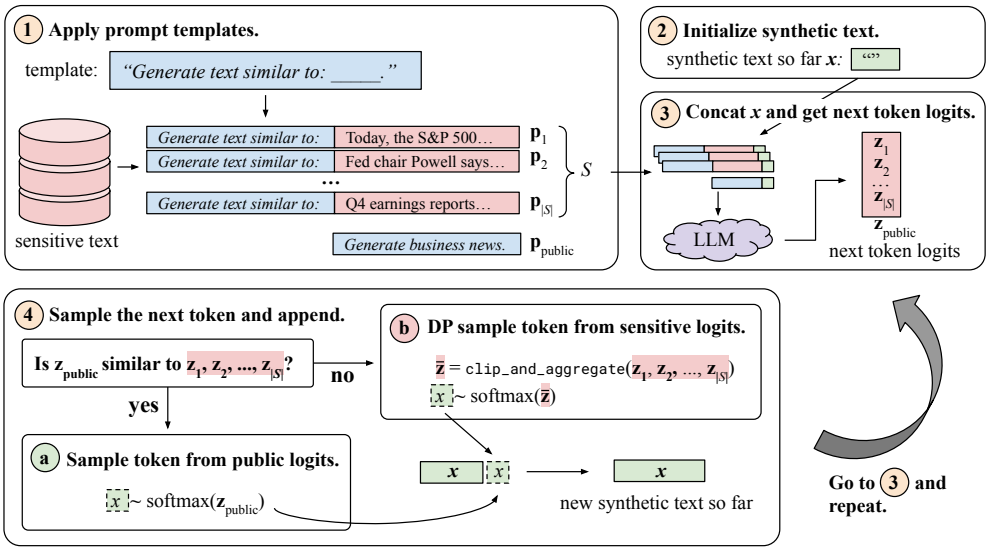


Figure 1: Algorithm 1, visualized. An LLM receives a batch of prompts, each instructing to generate text similar to a piece of sensitive text. *Synthetic text* is generated token by token, by running inference on the batch in parallel. In each step, the logit vectors produced downstream of sensitive text are aggregated and sampled from with differential privacy. Every token sampled in such way incurs a privacy cost, motivating us to include an auxillary public prompt and sample from its logits when similar to the sensitive logits.

1.1 Our contributions

In this paper we describe a new private prediction method that produces hundreds of times as much synthetic data as a state-of-the-art private prediction method, while maintaining a comparable privacy guarantee. Similar to some existing work, our method is based on running LLM inference on several subsets of the input data in parallel and privately aggregating their token distributions to generate synthetic text. However, our approach is distinguished by three novel algorithmic elements that lead to its improved performance:

1. Better private token selection. Instead of protecting the privacy of the entire token distribution with Gaussian or Laplace noise, we leverage the uncertainty inherent in sampling to ensure privacy. We clip and aggregate token logits before standard softmax sampling — which is differentially private, since it can be viewed as the exponential mechanism. Our approach induces much less distortion of the original token distributions to achieve the same level of privacy than prior work.

2. Avoiding prefix re-sampling. Prior work generated each token using a random subset of the input data, leveraging privacy amplification by subsampling in their analysis. This is computationally undesirable, as it requires repeated re-computation of the prefix for each decoding step, thus limiting scalability towards generating large synthetic corpora. Indeed, prior work describes this re-sampling as the "main weakness" of the approach (Tang et al., 2024). To resolve the problem, we instead generate each synthetic example using a fixed disjoint subset of the input data, which yields substantial savings in privacy cost – via *parallel composition* – while allowing us to pay linear instead of quadratic non-attention FLOPs in terms of sequence length via KV cache accelerated decoding.

3. Leveraging public predictions. Our method uses an auxillary token distribution from an LLM without access to sensitive data, and draws the next token from that distribution whenever it is very similar to the token distribution induced by the sensitive data. Our method incurs no privacy cost when outputting "obvious" tokens, and as a result, only a fraction of the tokens in the synthetic data are generated using sensitive data (as little as 20% in structured datasets). We leverage the sparse vector technique to privately calculate distributional similarity.

Taken together, the combination of these algorithmic techniques leads to significant improvements over prior work. Roughly speaking, (1) and (2) above keep our inference closely aligned to standard (non-DP) inference.

In our experiments, we generate private synthetic versions of publicly available, benchmark machine learning datasets, and then use the synthetic datasets for downstream classification and ex-

traction tasks. Owing to the increased quantity and quality of our synthetic data, we improve over an existing state-of-the-art private prediction method in terms of downstream accuracy. Furthermore, while prior work in this paradigm only generated a small (<10) number of examples, we demonstrate the ability to generate thousands of training examples, enough for fine-tuning downstream models.

Finally, since synthetic data is intended for a wide variety of applications, we also evaluate data quality using a metric that is orthogonal to performance on downstream tasks. Specifically, we generate synthetic versions of a publicly available dataset containing highly structured data records, each of which is encoded as a JSON object. Our results demonstrate that the sparse vector technique helps preserve data structure at high privacy levels.

2 Related work

Private fine-tuning is widely used for synthetic text generation. Yue et al. (2023) created private synthetic versions of text datasets by using DP-SGD (Abadi et al., 2016) to fine-tune an LLM on the sensitive data. Kurakin et al. (2024) showed that parameter efficient approaches to fine-tuning, such as LoRA (Hu et al., 2022) can improve the quality of the synthetic data. Wu et al. (2024a) took a two-stage approach: First they fine-tuned an LLM on a public dataset that closely resembled the sensitive data (which was itself generated by an LLM using carefully designed prompts); then they completed the fine-tuning process by running DP-SGD on the sensitive dataset. Concurrent to the present work, Tran and Xiong (2024) describe a private fine-tuning approach for generating synthetic tabular data that is formatting compliant.

Private prediction (Dwork and Feldman, 2018) is an alternate approach to private machine learning that only guarantees the privacy of the predictions output by an ML model, and not the model itself. The predominant way this is realized is via *subsample-and-aggregate* (Nissim et al., 2007): First sensitive data is split into disjoint partitions; then non-private predictions are made from each partition and privately aggregated. PATE (Papernot et al., 2017, 2018) employs this approach to get answers to a limited set of image classification queries, which are then used to train a student model that can be queried indefinitely.

Private prediction has been applied to synthetic text generation by viewing each token sampled by an LLM as a 'prediction', and perturbing the LLM's token distributions to ensure their privacy. Tang et al. (2024) added noise to several independent token distributions and averaged them, while Hong et al. (2024) selected the most popular token among the token distributions using the Limited-Domain mechanism (Durfee and Rogers, 2019). These methods can avoid the time, compute, and access required to fine-tune an LLM with billions of parameters. However, a privacy loss is suffered for each *token* produced in this manner. As a result, previous work has only been able to generate a very small number of synthetic examples at reasonable privacy levels (fewer than 10). Other work has applied private prediction techniques to LLMs (Majmudar et al., 2022; Duan et al., 2023), including in combination with fine-tuning (Ginart et al., 2022; Flemings et al., 2024), but not for the purpose of synthetic text generation.

Finally, another distinct set of approaches are *private filtering* methods. Private filtering methods operate directly on whole LLM responses and a large corpus of public data that does not require protection. Yu et al. (2024) and Xie et al. (2024) used the sensitive responses to privately select similar responses from the public dataset. Similarly, Wu et al. (2024b) aggregate response embeddings and select the public response that is closest in embedding space.[1] One limitation of filtering methods is that the menu of possible responses is constructed without signal from the new source dataset.

3 Method

3.1 Standard LLM inference

Before describing our algorithm for generating private synthetic text, we review the standard algorithm for LLM inference. Let $\mathcal{X}$ be the token vocabulary (*i.e.*, the set of all possible tokens), and let $v = |\mathcal{X}|$ be the vocabulary size. A *token sequence* is an element of $\mathcal{X}^*$, and a *logit vector* is an element of $\mathbb{R}^v$ (one logit per token in the vocabulary). If $\mathbf{x}_1$ and $\mathbf{x}_2$ are token sequences then we write $\mathbf{x}_1\mathbf{x}_2 \in \mathcal{X}^*$ to denote their concatenation.

A decoder-only LLM can be viewed as a function $\text{logits} : \mathcal{X}^* \to \mathbb{R}^v$ that maps each token sequence to a logit vector. Standard LLM inference generates a *response* $\mathbf{x} \in \mathcal{X}^*$ by initializing $\mathbf{x} = \mathbf{p}$, where $\mathbf{p} \in \mathcal{X}^*$ is the *prompt*,

[1] Wu et al. (2024b) also proposes a non-filtering approach based on privately selecting common keywords among the sensitive data and using them to prompt an LLM.

and then repeatedly executes the following procedure: (1) Let $\mathbf{z} = \text{logits}(\mathbf{x})$; (2) draw token x from $\text{softmax}(\mathbf{z}/\tau)$; and (3) append x to $\mathbf{x}$. Here $\text{softmax}(\mathbf{z}/\tau)$ is the distribution that assigns probability proportional to $\exp(z_i/\tau)$ to the ith token, and $\tau > 0$ is a *temperature* parameter that flattens or sharpens the distribution. The procedure terminates when $x =$ <eos>, a special token that indicates the end of the response.

3.2 Our algorithm

One straightforward approach to generating a synthetic version of a sensitive piece of text would be to prompt an LLM with '*Please generate text similar to:* <*sensitive text*>'. However, this could easily lead to a privacy violation, as the response could retain the semantics of the input sensitive text.

Algorithm 1 describes our method for privately generating a dataset of synthetic examples X from a dataset of sensitive prompts D. Each prompt in D resembles the sample prompt given above. But instead of using a *single* prompt to generate a synthetic example, Algorithm 1 takes a *batch* of prompts S and runs LLM inference in parallel on each prompt. A synthetic example is generated one token at a time, with the average of the logit vectors across the batch defining the distribution from which the next token is randomly selected. Before averaging, a logit vector $\mathbf{z}$ is clipped and re-centered using the function

$$\text{clip}_c(\mathbf{z})_i = \max\{-c, \mathbf{z}_i - \max_j \{\mathbf{z}_j\} + c\} \quad (1)$$

which maps each component i of $\mathbf{z}$ into the target clipping range $[-c, c]$. Forcing each logit to lie in a bounded range is key to proving the privacy guarantee for our algorithm (see §4). While several functions can achieve this purpose, Eq. (1) has an additional desirable property: If the components of $\mathbf{z}$ can be shifted by a constant so that they all lie in the interval $[-c, c]$, then $\text{clip}_c(\mathbf{z})$ is one such shift. This property is desirable because the distribution $\text{softmax}(\mathbf{z})$ is invariant to any constant shift of $\mathbf{z}$. Empirically, we found that Eq. (1) performed better than other functions considered. For example, regular clipping to the range $[-c, c]$ without recentering requires twice as large c to sample without distortion (see §B).

Since the average logit vector is computed using sensitive prompts, each token selected from a distribution determined by the average logit vector incurs a privacy cost. To minimize this cost, Algorithm 1 also has access to a non-sensitive public prompt, $\mathbf{p}_{\text{public}}$, and uses this prompt to generate the next token whenever doing so does not significantly change the distribution from which the next token is drawn. The distance function used to make this determination is

$$\text{d}(Z, \mathbf{z}_{\text{public}}) = \|\frac{1}{s}\sum_{\mathbf{z}\in Z} p_{\mathbf{z}} - p_{\mathbf{z}_{\text{public}}}\|_1, \quad (2)$$

where $p_{\mathbf{z}} := \text{softmax}(\mathbf{z})$, Z are the logit vectors computed for each sensitive prompt in S, $\mathbf{z}_{\text{public}}$ is the logit vector computed using $\mathbf{p}_{\text{public}}$, and s is the expected batch size. When this distance is small, Algorithm 1 outputs a public token instead of a private token. The privacy guarantee for Algorithm 1 leverages the analysis of the sparse vector technique (Dwork et al., 2009), and shows that while privacy degrades with the number of private output tokens, it is independent of the number of public output tokens (see §4). Empirically, we observe that the fraction of output tokens that must be pri-

Algorithm 1 Generate private synthetic examples

Parameters: LLM $\text{logits}(\cdot)$, public prompt $\mathbf{p}_{\text{public}}$, expected batch size s, private tokens to sample r. *Sampling*: clipping bound c, temperature τ, public temperature τ_{public}. *Public optionality*: public/private distribution distance $\text{d}(\cdot,\cdot)$, threshold θ, noise level σ.
Input: Dataset of sensitive prompts D; each prompt contains a sensitive example
Output: Dataset of synthetic examples X

$X \leftarrow \emptyset$
Let $\mathcal{S}$ be a partition of D into disjoint batches
for each batch $S \in \mathcal{S}$ **do**
  $\hat{\theta} \leftarrow \theta + \text{Laplace}(\sigma)$ `# Init noisy threshold`
  $t \leftarrow 0$ `# Private token counter`
  while $t < r$ **do**
    $\mathbf{x} \leftarrow$ Empty token sequence
    while $\mathbf{x}$ does not end with <eos> **do**
      $Z \leftarrow \{\text{logits}(\mathbf{px}) : \mathbf{p} \in S\}$
      $\mathbf{z}_{\text{public}} \leftarrow \text{logits}(\mathbf{p}_{\text{public}}\mathbf{x})$
      `# Check if pub/priv distributions are far`
      $\hat{d} \leftarrow \text{d}(Z, \mathbf{z}_{\text{public}}) + \text{Laplace}(2\sigma)$
      if $\hat{d} \geq \hat{\theta}$ **then** `# Sample priv token`
        $\bar{\mathbf{z}} \leftarrow \frac{1}{s}\sum_{\mathbf{z}\in Z} \text{clip}_c(\mathbf{z})$
        $x \sim \text{softmax}(\bar{\mathbf{z}}/\tau)$
        $t \leftarrow t + 1$
        $\hat{\theta} \leftarrow \theta + \text{Laplace}(\sigma)$
      else `# Sample pub token`
        $x \sim \text{softmax}(\mathbf{z}_{\text{public}}/\tau_{\text{public}})$
      Append x to $\mathbf{x}$
    $X \leftarrow X \cup \{\mathbf{x}\}$
return X

vate in order to generate high-quality synthetic data can be as low as 20% for highly structured datasets.

Note that the first step of Algorithm 1 partitions the input dataset of sensitive prompts into disjoint batches. We do not prescribe a procedure for assigning prompts to batches in Algorithm 1 since many batching approaches are admissible as long as they satisfy a minor technical assumption required for the privacy analysis of Algorithm 1, which we explain in §4. While the batches are not required to be any particular size, the algorithm runs faster if each batch has size equal to the expected batch size s. And while prompts can be batched together (almost) arbitrarily, more tailored batching can lead to better synthetic data quality. For example, in the experiments in §5, where we generate synthetic versions of ML training datasets, each sensitive prompt contains a label. In those experiments we assign prompts with the same label to the same batch.

3.3 Comparison to prior algorithms

Two major features of Algorithm 1 are that it leverages the inherent randomness of token sampling to guarantee privacy, and that it further reduces privacy cost by using public data to generate a portion of the synthetic data. Some prior work also incorporated these algorithmic ideas, but with key differences. Instead of clipping logits to ensure that the token sampling is private, Majmudar et al. (2022) mixed each sensitive token distribution with the uniform distribution. This approach induced a dependence on the vocabulary size in their privacy guarantee, and since LLM vocabularies are typically very large, the resulting privacy guarantee was quite weak: Majmudar et al. (2022) noted that setting the differential privacy parameter ε lower than 50 produced synthetic data that was "unusable". Flemings et al. (2024) guaranteed the privacy of token sampling by mixing each sensitive token distribution with a public token distribution, but their approach was based on aggregating a set of fine-tuned models, not a set of prompts. Neither Majmudar et al. (2022) nor Flemings et al. (2024) aim to generate synthetic data.

Tang et al. (2024) found that limiting the token vocabulary to a fixed set of the most popular 100 public tokens caused their synthetic data generation algorithm to exhibit greater stability. However, if the sensitive data contains many tokens that are rare in public data, their approach cannot produce synthetic data that is very similar to the sensitive data. By contrast, our approach compares public and private token distributions on-the-fly, and determines which one to use for sampling the next token by balancing a trade-off between privacy and quality. Also, Tang et al. (2024) used a different random subset of prompts to generate each token, and left as an open problem how to use a single subset to generate every token in a synthetic example. Our algorithm resolves this open problem, and consequently yields both improved privacy and greater computational efficiency (see §6).

4 Privacy analysis

In this section we state formally how Algorithm 1 preserves the privacy of the sensitive prompts it uses to generate synthetic examples.

Let $\mathcal{D}$ be the set of all possible prompt datasets. A *mechanism* is a randomized function with domain $\mathcal{D}$. Note that Algorithm 1 is a mechanism. We say that a pair of prompt datasets $D, D' \in \mathcal{D}$ are *neighbors* if there exists a prompt $\mathbf{p}$ such that $D = D' \cup \{\mathbf{p}\}$ or $D' = D \cup \{\mathbf{p}\}$. In the differential privacy literature this is commonly referred to as the *add/remove* neighbor relation.

Definition 1 (Dwork et al. (2006))**.** *A mechanism M satisfies* (ε, δ)-differential privacy *if* $\Pr[M(D) \in O] \leq e^{\varepsilon} \Pr[M(D') \in O] + \delta$ *for any neighboring datasets $D, D' \in \mathcal{D}$ and subset O of the range of M.*

Theorem 1 below provides a differential privacy guarantee for Algorithm 1. The proof of Theorem 1 requires a technical assumption about how the prompts are partitioned into batches in the first step of the algorithm.

Assumption 1. *In Algorithm 1, the assignment of a prompt to a batch depends only on the prompt itself, and not on the other prompts.*

The most straightforward way to satisfy Assumption 1 is to apply a hash function to each prompt and then use the hash value to determine its assigned batch. For example, if h is the hash value, n is the number of prompts and s is the expected batch size, then we can assign the prompt to the $(h \mod \frac{n}{s})$th batch. If we want to batch together prompts that share a certain attribute (like a label), we can apply another hash function to that attribute and concatenate the hash values. Using hash functions for batch assignment can lead to batches whose sizes differ from the expected batch size s, but this does not impact the validity of Theorem 1.

Theorem 1 (Privacy of Algorithm 1). *Suppose Assumption 1 holds. Let* $\rho = r\left(\frac{1}{2}\left(\frac{c}{s\tau}\right)^2 + \frac{2}{(s\sigma)^2}\right)$. *For all* $\varepsilon \geq 0$, *Algorithm 1 satisfies* (ε, δ)*-differential privacy, where*

$$\delta = \inf_{\alpha \in (1,\infty)} \frac{e^{(\alpha-1)(\alpha\rho-\varepsilon)}}{\alpha - 1}\left(1 - \frac{1}{\alpha}\right)^{\alpha}.$$

Also, for all $\delta \in (0, 1]$, *Algorithm 1 satisfies* (ε, δ)*-differential privacy, where*

$$\varepsilon = \rho + \sqrt{4\rho \log(1/\delta)}.$$

The proof is in §C and makes use of sharp privacy analyses of: (1) zCDP to approximate DP conversion (Canonne et al., 2020); and (2) zCDP bounds for the exponential mechanism (Cesar and Rogers, 2021).

5 Experiments

Gemma 1.1 2B IT (Gemma Team, 2024) is the data generator in our main private prediction experiments. We choose it due to its lightweight, open-source JAX implementation[2] that makes easy to implement and share sampling algorithms. Tables 1a and 1b give an overview of datasets and models used.

Dataset	n_{train}	Description
AGNews	120,000	4-way news classification
TREC	5452	6-way query classification
DBPedia	560,000	14-way topic classification
MIT-G	2,953	Movie genre extraction
MIT-D	1,561	Movie director extraction
IMDB	25,000	2-way review classification
Yelp	560,000	2-way review classification
WikiMoviesJSON	27,412	JSON with 6 fields

(a) Overview of datasets used.

Model	Usage
Gemma 1.1 2B IT	Generation; private prediction
LaMDA 8B	Generation; DP fine-tuning
GPT-3 babbage-002	Evaluation; in-context learning
BERT-Base 12/768 110M	Evaluation; fine-tuning

(b) Overview of models used in main experiments.

Table 1: Overview of datasets and models used in our main experiments. Datasets are benchmark classification and extraction tasks used in prior work on private synthetic text generation, with the exception of WikiMoviesJSON, which is used for structured data experiments. LaMDA and Gemma are used for synthetic data generation, while the other models are used to measure how useful our synthetic data is for improving accuracy on real test data.

We perform 3 sets of experiments, targeting various datasets and utility criteria:

[2] https://github.com/google-deepmind/gemma

In-context learning (§5.1). We generate examples to use as in-context exemplars for prompting an LLM. We report downstream accuracy on real test examples, when prompted with synthetic data, on 3 classification tasks (*AGNews* (Zhang et al., 2015), *DBPedia* (Zhang et al., 2015), *TREC* (Voorhees and Tice, 2000)) and 2 extraction tasks (*MIT-G*, *MIT-D* (Liu et al., 2012)).

Fine-tuning (§5.2). We generate synthetic examples to use for fine-tuning a BERT classifier. We report downstream accuracy on real test examples for 3 classification tasks (*IMDB* (Maas et al., 2011), *Yelp* (Zhang et al., 2015), *AGNews* (Zhang et al., 2015)).

Structured data (§5.3). We generate examples that must adhere to structural constraints to be useful synthetic data. We consider a JSON generation task (*WikiMoviesJSON* (Rust, 2024)), evaluating structure preservation.

5.1 In-context learning

Experimental setup. Using our method, we generate 90-1500 examples using Gemma 1.1 2B IT. We compare against real examples, and results reported in the prior work of Tang et al. (2024), where they generated 4-shot examples for in-context learning.[3] To evaluate generated synthetic data, we put synthetic examples in the context window before querying with the real test example, as shown in Figure 2.

```
Classify the following examples:
Text: lorem ipsum   # synthetic text 1
Answer: label
# ...
Text: sed do eiusm  # synthetic text n
Answer: label

Text: excepteur si  # test text
Answer:
```

Figure 2: Example of n-shot in-context learning evaluation for synthetic data.

Results. Results are presented in Table 2. Our gains in quantity while maintaining quality are realized in terms of 64-shot in-context learning accuracy. In some cases, we can generate more examples, but we limit ourselves to 64 for these

[3] It is no longer possible to reproduce their results, due to changes in the OpenAI API since publication: GPT-3 babbage is now deprecated, and it is no longer possible to query for top 100 logprobs, which is required by their method.

ε	Method	Shots	Reported in	Model	GPT-3 babbage-002 Acc. (%)*				
					AGNews	DBPedia	TREC	MIT-G	MIT-D
0	Zero shot	0	This work	-	$24.8_{0.0}$	$12.0_{0.0}$	$28.4_{0.0}$	$29.6_{0.0}$	$28.8_{0.0}$
∞	Real data	4	This work	-	$75.3_{3.0}$	$73.6_{0.3}$	$34.9_{5.0}$	$56.0_{2.0}$	$83.1_{5.3}$
		64			$84.7_{1.5}$	$92.5_{1.6}$	$50.3_{6.1}$	$56.4_{5.4}$	$89.1_{0.7}$
	Tang et al. (2024)	4	Tang et al. (2024)*	GPT-3 babbage	$69.3_{4.8}$	$82.3_{3.7}$	$50.6_{6.9}$	$54.4_{7.0}$	-
	Ours	4	This work	Gemma 1.1 2B IT	$76.8_{4.8}$	$72.3_{2.5}$	$38.8_{6.0}$	$47.7_{2.5}$	$81.7_{2.4}$
		64			$77.5_{1.8}$	$91.5_{1.7}$	$57.9_{3.4}$	$56.4_{1.2}$	$87.1_{0.2}$
1	Tang et al. (2024)	4	Tang et al. (2024)*	GPT-3 babbage	$64.1_{3.9}$	$81.2_{3.0}$	$50.7_{4.1}$	$46.3_{7.8}$	$69.2_{7.9}$
		4	This work	Gemma 1.1 2B IT	$74.9_{3.8}$	$80.9_{3.6}$	$36.7_{2.2}$	$34.1_{9.3}$	$78.7_{1.9}$
	Ours	4	This work	Gemma 1.1 2B IT	$75.9_{3.5}$	$75.1_{0.5}$	$39.2_{3.7}$	$47.1_{6.0}$	$84.5_{1.0}$
		64			$78.7_{1.8}$	$90.4_{2.6}$	$53.6_{1.3}$	$51.6_{2.3}$	$86.4_{0.6}$

Table 2: In-context learning results with GPT-3 babbage-002. We report mean and standard deviation over 3 random samplings (equally many from each label for classification; fully random for extraction) of synthetic/real data. **(*) Note**: For the results reported in Tang et al. (2024), they use GPT-3 babbage (now deprecated; we use GPT-3 babbage-002) as the downstream in-context learner, and use the top 100 logprobs for contextual calibration (only top 5 are available now). While not directly comparable, we report their results for context.

evaluations for cost and efficiency reasons. Our results at 64 shots are comparable to real data at 64 shots. Notably, our synthetic data at 64 shots improves over real data at 4 shots – a rough upper bound on the performance of methods limited to generating 4 examples (e.g., Tang et al. (2024)). We also improve over results reported in Tang et al. (2024) – however as there are differences in the experimental setup, we also report the results of our re-implementation.[4]

We evaluate with GPT-3 babbage-002 which has a 16K context window. We report results on *AGNews*, *DBPedia*, *TREC*, *MIT-G*, and *MIT-D* using the implementation of Zhao et al. (2021). Following the work of Tang et al. (2024), we enable contextual calibration (Zhao et al., 2021) for classification but not extraction tasks. Our evaluation setup is a best-effort reproduction of their setup, which is no longer possible to completely reproduce due to changes to OpenAI API access (see Table 2 caption). Due to cost, we follow prior work (Bertsch et al., 2024; Ratner et al., 2023; Lu et al., 2022; Zhao et al., 2021) and opt to subsample test sets to 250 test examples. We run 3 seeds of sampling of exemplars from synthetic/real data. Additionally, we present a limited set of results on Gemma 2 2/9/27B IT, studying the effect of model size on classification performance in §A.2.

5.2 Fine-tuning

We achieve significant improvements over the best available private inference method for in-context learning tasks. Since our method is capable of generating thousands of synthetic examples at reasonable privacy budgets, it is natural to ask whether it can compete with state-of-the-art private fine-tuning methods, which can generate infinitely many synthetic examples once the up-front costs of model training are paid. This makes them capable of producing enough data to train downstream classification models.

Experiment setup. We use our approach to generate a large quantity of synthetic data for the purposes of fine-tuning 110M BERT-Base models. We consider 3 classification tasks used in prior work on private fine-tuning (Kurakin et al., 2024)), following the exact same evaluation procedure. We omit comparison to prior private prediction work (e.g. (Tang et al., 2024)), as they only generate 4 examples which is insufficient for fine-tuning.

Results. Main results are presented in Table 3. Across various datasets and privacy levels, we generate between 2.5K (IMDB, $\varepsilon = 1$) and 200K (Yelp, $\varepsilon = 10$) examples for fine-tuning. Prior work generating fewer than 10 examples using private prediction were unable to compare with private fine-tuning on these tasks. While there remains a gap between the best fine-tuning and best private inference methods on downstream classification tasks, we achieve reasonable performance, even out-performing or matching the baseline of privately tuning all the parameters in the model reported in Kurakin et al. (2024).

Limited data regime. We additionally consider the limited data regime. In §A.1 we present exper-

[4]Specifically, we use their best hyperparameters (from Appendix E, Table 9 of (Tang et al., 2024)) and algorithm, but with our model, prompt, and evaluation setup.

			BERT Acc. (%)											
			IMDB @ ε				Yelp @ ε				AGNews @ ε			
Method	Reported in	Model	∞	1	3	10	∞	1	3	10	∞	1	3	10
Real data	(Kurakin et al., 2024)	-	$93.7_{0.1}$	-	-	-	$97.6_{0.1}$	-	-	-	$93.7_{0.1}$	-	-	-
Fine-tune			$93.2_{0.2}$	$79.1_{1.7}$	$83.9_{0.6}$	$84.0_{0.7}$	$95.9_{0.1}$	$84.1_{0.3}$	$84.6_{0.1}$	$84.2_{0.3}$	$91.1_{0.1}$	$65.7_{2.9}$	$65.3_{2.7}$	$65.1_{5.3}$
Prompt-tune	(Kurakin et al., 2024)	LaMDA 8B	$92.0_{0.1}$	$88.1_{0.4}$	$87.4_{0.2}$	$90.7_{0.2}$	$93.9_{0.1}$	$94.1_{0.1}$	$93.5_{0.1}$	$94.1_{0.1}$	$88.3_{0.3}$	$83.9_{0.8}$	$86.2_{0.2}$	$86.9_{0.1}$
LoRA			$91.6_{0.2}$	$90.0_{0.3}$	$90.6_{0.2}$	$91.3_{0.2}$	$96.4_{0.1}$	$95.5_{0.1}$	$95.6_{0.1}$	$95.9_{0.1}$	$91.8_{0.2}$	$89.4_{0.1}$	$89.6_{0.1}$	$90.0_{0.1}$
Ours	This work	Gemma 1.1 2B IT	$83.6_{2.9}$	$82.7_{2.1}$	$83.6_{1.9}$	$85.5_{2.3}$	$91.8_{0.6}$	$91.1_{0.2}$	$91.6_{0.8}$	$92.6_{0.2}$	$81.2_{1.2}$	$79.8_{1.8}$	$79.3_{2.1}$	$79.8_{0.3}$
+ SVT	This work	Gemma 1.1 2B IT	-	$84.3_{1.1}$	$84.4_{1.5}$	$85.0_{1.0}$	-	$88.4_{0.6}$	$89.1_{0.3}$	$89.0_{1.9}$	-	$79.2_{0.3}$	$79.8_{0.4}$	$80.4_{0.6}$

Table 3: Results of fine-tuning on real and synthetic data with BERT. We report mean and standard deviation over 3 runs of downstream fine-tuning and evaluation. We compare to results reported in (Kurakin et al., 2024) that fine-tunes a synthetic data generator with DP-SGD. We generate 2.5-200K examples with private prediction, which suffices to train reasonably performing models on.

iments on *AGNews1K*, a 1024-subsample of *AGNews*. Our method, which employs parallel composition, is "pay-as-you-go", i.e., we can put in a small amount of data to get out a small amount, while preserving quality. On the other hand, fine-tuning based approaches necessarily pay upfront to ensure the model and all future generations are private. This means that without sufficient data, all outputs will be low quality. Results in Table 5 demonstrate that our private prediction method generates more useful examples for in-context learning in this limited data regime.

5.3 Structured data

We conclude our experiments with a demonstration of the lift in performance provided by using the sparse vector technique (SVT) against a public prompt. Informally, the privacy loss of our method only scales with the information density of a new example *vis-a-vis* the public prompt. This contrasts with other private inference methods that incur privacy loss on every token. This is especially useful for structured data, where we avoid incurring privacy loss on syntactic elements of the data.

Experiment setup. For JSON generation, we evaluate on a dataset of information about American movies scraped from Wikipedia (Rust, 2024). Entries contain fields such as `title`, `year`, `cast`, and `extract` (a short synopsis). We lightly curate the data: we omit uninteresting fields (i.e., thumbnail dimensions and URLs) and remove entries with incomplete entries. We refer to the resulting 34,266 JSON examples with 6 fields as *WikiMoviesJSON*. We evaluate two criteria: the rate at which output generated constitutes well-formed JSON (*Parses (%)*), and rate at which the output passes basic schema validation (*Validates (%)*). This includes checks such as: no extra fields, all required fields are present, values are the correct type, and other custom constraints (e.g. no whitespace in the `href` field).

ε	Method	τ	Parses (%)	Validates (%)	m
1	Ours	2	$80.6_{1.3}$	$74.2_{1.9}$	$94.3_{1.2}$
		2.5	$4.9_{1.1}$	$1.5_{0.1}$	$138.0_{7.5}$
	+ SVT, $\theta = 0.9$	2	$91.7_{2.1}$	$88.6_{3.2}$	$289.7_{19.4}$
		2.5	$74.1_{2.7}$	$64.0_{4.1}$	$356.7_{25.9}$
	+ SVT, $\theta = 1.5$	2	$95.5_{1.0}$	$93.1_{0.7}$	$893.0_{20.2}$
		2.5	$79.3_{1.0}$	$72.7_{1.4}$	$1178.3_{10.1}$

Table 4: Results for generating JSON records from *WikiMoviesJSON*. We report mean and standard deviation over 3 runs of dataset generation. τ refers to the sampling temperature, and m refers to the number of raw samples produced (before parsing and validation checks). The batch size used is 255. We present results at two different SVT thresholds θ, and see gains in structure preservation and quantity.

Results. Results are in Table 4. Targeting a large number of examples at small ε necessitates increases in the sampling temperature τ, to ensure privacy, but compromises the well-formed-ness of outputs. For structured generation, there is a large amount of tokens that (a) are crucial to get right for structure preservation, and (b) easily predictable without access to sensitive data. Here the SVT enables us to get these tokens reliably and for free, leading to better generation quantity.

6 Discussion

We believe that our significantly improved performance relative to Tang et al. (2024) is primarily attributable to two algorithmic innovations.

First, for each generated token, Tang et al. (2024) preserve the privacy of the entire distributions from which the token is sampled (by taking argmax), even though only the token itself is included in the synthetic data. By contrast, our method uses a discrete choosing mechanism, the exponential

mechanism. As a result, we do not need to maintain a DP version of the entire token distribution to release a single token. This decision leads to significantly lower noise requirements, as a straightforward calculation reveals. Empirically, we obtained good synthetic data quality with $s = 250$, $\tau = 2$, $c = 10$ and $\delta = 10^{-6}$. In order to switch to the Gaussian mechanism using its standard (ε, δ)-DP guarantee, and achieve comparable privacy guarantees we would would require $\sigma \approx 0.53$ to achieve a comparable privacy guarantee. (See §D). Better analyses of the Gaussian mechanism exist, but do not offer much help. Using the improved analysis in Balle and Wang (2018) to attain the same ε would require $\sigma \approx 0.34$. Conducting the analysis so that both mechanisms have equivalent privacy loss under zCDP yields $\sigma = 0.2$. These are all very large noise magnitudes relative to probabilities in $[0, 1]$.[5]

Secondly, Tang et al. (2024) generated each token using a different random sample of the sensitive prompts, which is computationally very expensive, as it prevents the use of KV cache-accelerated decoding, since the cache is invalidated upon every resample. While resampling less often would be more practical, Tang et al. (2024) noted that in this case the privacy amplification benefits of subsampling would not be adequately realized, and characterized this limitation as the "main weakness" of their approach. Instead, our method generates each synthetic example using a fixed disjoint subset of the sensitive prompts, allowing us to leverage parallel composition in our analysis, and thus avoid this privacy versus computation tradeoff.

7 Conclusion

As proprietary models become increasingly powerful, we anticipate more practitioners will be able to generate inferences from state-of-the-art models, while fewer practitioners will be able to *train* networks that perform like state-of-the-art models. This makes it increasingly important to develop private prediction methods that compete with private fine-tuning.

We demonstrate that private prediction can be used to generate large amounts of synthetic text with reasonable differential privacy guarantees. We produce 2-3 orders of magnitude more private synthetic data than what was demonstrated in prior work in this paradigm. Access to more synthetic data lets us fine-tune downstream models, as well as yields performance improvements via many-shot in-context learning. Furthermore, we introduce a novel use of public models in which we are able to sample predictable tokens at no privacy cost, which is particularly effective for structured data.

Limitations

While our work demonstrates that private prediction is a practical technique for privately generating a large volume of high-quality synthetic data, there remains a gap between our results and the results obtained from privately fine-tuning the parameters of the LLM. Currently, private prediction methods pay a privacy cost for every generated token, while private fine-tuning methods do not. Finally, any method for ensuring data privacy will inevitably entail some loss of data utility.

Author contributions

- **Alex B** is the main contributor. He implemented the method, tested variants to optimize utility and privacy, and ran most of the experiments. He also proposed the use of sparse vector.
- **Umar** proposed the method, the use of sampling to preserve privacy, and conducted the theoretical analysis.
- **Umar** and **Kareem** framed the structure of the paper and led writing.
- **Kareem** proposed parallel composition. He also assisted with the privacy analysis.
- **Natalia** proposed logits recentering.
- **Weiwei** and **Alexey** provided infrastructure support and code for running experiments. **Alexey** suggested the limited data experiments and ran the fine-tuning baselines.
- **Natalia**, **Andreas**, and **Sergei** advised the project.
- **Everyone** contributed to discussing, interpreting, and iterating on experiment results as well as project management.

[5] To put independent noise of magnitude $\sigma = 0.2$ into perspective: suppose the ground truth next-token prediction is deterministic, i.e., $\bar{\mathbf{p}} = [1, 0, ..., 0] \in \mathbb{R}^v$, v = 256128 in the case of Gemma. Now with probability ≥ 0.15, the noised distribution $\widetilde{\mathbf{p}}$ has $\widetilde{\mathbf{p}}_1 < 0.8$. Each other $\mathbf{p}_i$ is ≥ 0.8 w.p. $\geq 3 \cdot 10^{-5}$ independently. Hence the probability of one of these being promoted to argmax is $\geq 0.15 \cdot (1 - (1 - 3 \cdot 10^{-5})^{v-1}) \approx 0.15$. At this rate, the chance of generating a 30 token span without a corruption is $< 1\%$.

References

Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318.

Borja Balle and Yu-Xiang Wang. 2018. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In *International Conference on Machine Learning*, pages 394–403. PMLR.

Amanda Bertsch, Maor Ivgi, Uri Alon, Jonathan Berant, Matthew R Gormley, and Graham Neubig. 2024. In-context learning with long-context models: An in-depth exploration. *arXiv preprint arXiv:2405.00200*.

Mark Bun and Thomas Steinke. 2016. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*, pages 635–658. Springer.

Clément L Canonne, Gautam Kamath, and Thomas Steinke. 2020. The discrete gaussian for differential privacy. *Advances in Neural Information Processing Systems*, 33:15676–15688.

Mark Cesar and Ryan Rogers. 2021. Bounding, concentrating, and truncating: Unifying privacy loss composition for data analytics. In *Proceedings of the 32nd International Conference on Algorithmic Learning Theory*, volume 132 of *Proceedings of Machine Learning Research*, pages 421–457. PMLR.

Haonan Duan, Adam Dziedzic, Nicolas Papernot, and Franziska Boenisch. 2023. Flocks of stochastic parrots: Differentially private prompt learning for large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 76852–76871. Curran Associates, Inc.

David Durfee and Ryan M Rogers. 2019. Practical differentially private top-k selection with pay-what-you-get composition. *Advances in Neural Information Processing Systems*, 32.

Cynthia Dwork and Vitaly Feldman. 2018. Privacy-preserving prediction. In *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 1693–1702. PMLR.

Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our data, ourselves: Privacy via distributed noise generation. In *Advances in Cryptology-EUROCRYPT 2006: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28-June 1, 2006. Proceedings 25*, pages 486–503. Springer.

Cynthia Dwork, Moni Naor, Omer Reingold, Guy N Rothblum, and Salil Vadhan. 2009. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 381–390.

James Flemings, Meisam Razaviyayn, and Murali Annavaram. 2024. Differentially private next-token prediction of large language models. *Preprint*, arXiv:2403.15638.

Gemma Team. 2024. Gemma: Open models based on gemini research and technology. *Preprint*, arXiv:2403.08295.

Antonio Ginart, Laurens van der Maaten, James Zou, and Chuan Guo. 2022. Submix: Practical private prediction for large-scale language models. *CoRR*, abs/2201.00971.

Junyuan Hong, Jiachen T. Wang, Chenhui Zhang, Zhangheng LI, Bo Li, and Zhangyang Wang. 2024. DP-OPT: Make large language model your privacy-preserving prompt engineer. In *The Twelfth International Conference on Learning Representations*.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Alexey Kurakin, Natalia Ponomareva, Umar Syed, Liam MacDermed, and Andreas Terzis. 2024. Harnessing large-language models to generate private synthetic text. *Preprint*, arXiv:2306.01684.

Jingjing Liu, Scott Cyphers, Panupong Pasupat, Ian McGraw, and James R. Glass. 2012. A conversational movie search system based on conditional random fields. In *INTERSPEECH 2012, 13th Annual Conference of the International Speech Communication Association, Portland, Oregon, USA, September 9-13, 2012*, pages 2454–2457. ISCA.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Jimit Majmudar, Christophe Dupuy, Charith Peris, Sami Smaili, Rahul Gupta, and Richard Zemel.

2022. Differentially private decoding in large language models. In *NAACL 2022 Second Workshop on Trustworthy Natural Language Processing (TrustNLP)*.

Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. 2007. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 75–84. ACM.

Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian J. Goodfellow, and Kunal Talwar. 2017. Semi-supervised knowledge transfer for deep learning from private training data. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. 2018. Scalable private learning with PATE. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Nir Ratner, Yoav Levine, Yonatan Belinkov, Ori Ram, Inbal Magar, Omri Abend, Ehud Karpas, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. Parallel context windows for large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6383–6402, Toronto, Canada. Association for Computational Linguistics.

Ryan Rogers and Thomas Steinke. 2021. A better privacy analysis of the exponential mechanism. DifferentialPrivacy.org. https://differentialprivacy.org/exponential-mechanism-bounded-range/.

Peter Rust. 2024. wikipedia-movie-data. https://github.com/prust/wikipedia-movie-data.

Xinyu Tang, Richard Shin, Huseyin A Inan, Andre Manoel, Fatemehsadat Mireshghallah, Zinan Lin, Sivakanth Gopi, Janardhan Kulkarni, and Robert Sim. 2024. Privacy-preserving in-context learning with differentially private few-shot generation. In *The Twelfth International Conference on Learning Representations*.

Toan V. Tran and Li Xiong. 2024. Differentially private tabular data synthesis using large language models. *Preprint*, arXiv:2406.01457.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962v2*.

Ellen M. Voorhees and Dawn M. Tice. 2000. Building a question answering test collection. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '00, page 200–207, New York, NY, USA. Association for Computing Machinery.

Shanshan Wu, Zheng Xu, Yanxiang Zhang, Yuanbo Zhang, and Daniel Ramage. 2024a. Prompt public large language models to synthesize data for private on-device applications. *Preprint*, arXiv:2404.04360.

Tong Wu, Ashwinee Panda, Jiachen T. Wang, and Prateek Mittal. 2024b. Privacy-preserving in-context learning for large language models. In *The Twelfth International Conference on Learning Representations*.

Chulin Xie, Zinan Lin, Arturs Backurs, Sivakanth Gopi, Da Yu, Huseyin A Inan, Harsha Nori, Haotian Jiang, Huishuai Zhang, Yin Tat Lee, Bo Li, and Sergey Yekhanin. 2024. Differentially private synthetic data via foundation model APIs 2: Text. In *ICLR 2024 Workshop on Secure and Trustworthy Large Language Models*.

Da Yu, Peter Kairouz, Sewoong Oh, and Zheng Xu. 2024. Privacy-preserving instructions for aligning large language models. *Preprint*, arXiv:2402.13659.

Xiang Yue, Huseyin Inan, Xuechen Li, Girish Kumar, Julia McAnallen, Hoda Shajari, Huan Sun, David Levitan, and Robert Sim. 2023. Synthetic text generation with differential privacy: A simple and practical recipe. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1321–1342, Toronto, Canada. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12697–12706. PMLR.

A Additional experiments

A.1 Private prediction beats fine-tuning in the limited data regime

We do LoRA fine-tuning with DP-SGD on *AGNews1K*, with the *same setup that beats our method in the full data regime*. We sample synthetic data from the fine-tuned model. We also run our private prediction method on *AGNews1K*. We evaluate performance on 4 and 16 shot in-context learning with GPT-3 babbage-002 (the same experimental setting as §5.1). Our private prediction approach outperforms the fine-tuning setup that does better in the full data regime.

ε	Method	Shots	Model	Acc. (%)
1	LoRA	4	LaMDA 8B	$63.3_{8.0}$
		16		$68.1_{5.9}$
	Ours	4	Gemma 1.1 2B IT	$73.9_{8.3}$
		16		$80.1_{2.5}$

Table 5: Results on *AGNews1K*, a 1024-subsample of AGNews. Our method is "pay-as-you-go", and is capable of generating a few high quality examples for in-context learning in this regime. On the other hand fine-tuning does worse due to the stricter requirement that all future model outputs must be private. Evaluation setup is the same in §5.1, except here we run 16 instead of 64-shot, because 16 examples produced by the LoRA model fills up the entire context length of babbage-002.

A.2 Effect of model size

We report results on the effect of the data generator's size on in-context classification performance on DBPedia. Our setup is the same as the experiments in §5.1, with the change that we use the Gemma 2 2/9/27B IT models to get more size variation in the same model family. This necessitated a slight change in the prompt (specifically, we append to the instruction: "`No formatting or explanations.`" For this evaluation, we see limited improvement due to scale.

ε	Shots	Model	Acc. (%)
1	4	Gemma 2 2B IT	$75.7_{0.8}$
	64		$91.2_{0.0}$
	4	Gemma 2 9B IT	$76.4_{1.2}$
	64		$92.4_{1.4}$
	4	Gemma 2 27B IT	$76.9_{2.2}$
	64		$91.9_{1.9}$

Table 6: Results on DBPedia classification. Evaluation setup is the same as in §5.1. We see limited improvement from the increase in model size.

B Design choices

B.1 Logits clipping function

In Figure 3, we compare results for different logits clipping functions. The baseline approach it to clip all logits to the interval $[-c, c]$ before aggregation and softmax – we refer to this as "fixed interval clipping". Alternatively, we can clip to the range $[\max_j\{\mathbf{z}_j\} - 2c, \max_j\{\mathbf{z}_j\}]$ and then translate to the interval $[-c, c]$ (Eq. 1). In Figure 3 we plot the distortion as a consequence of clipping in terms of L1 error, and find that the latter approach allows us clip more than twice as aggressively, thus improving the privacy guarantee, without compromising utility.

C Proof of Theorem 1

Our proof of Theorem 1 is organized into sections. §C.1 provides basic definitions. §C.2 and §C.3 establish key results related to composition and sensitivity. §C.4 proves the privacy of simpler mechanisms that each account for a portion of the functionality of Algorithm 1. C.5 puts all the pieces together and completes the proof.

C.1 Definitions

In §4 we defined neighboring prompt datasets. We extend the definition to arbitrary sets.

Definition 2. *Let $\mathcal{U}$ be a set. Let $S, S' \subseteq \mathcal{U}$. We say that S and S' are* neighbors *if there exists $u \in \mathcal{U}$ such that $S = S' \cup \{u\}$ or $S' = S \cup \{u\}$.*

The sensitivity of a function is an upper bound on how much its value can change over neighbors.

Definition 3. *Let $\mathcal{U}$ be a set. Let $k \geq 1$. Let $f : 2^{\mathcal{U}} \to \mathbb{R}^k$. The* sensitivity *of f is*

$$\sup_{S,S'} \left\| f(S) - f(S') \right\|_\infty$$

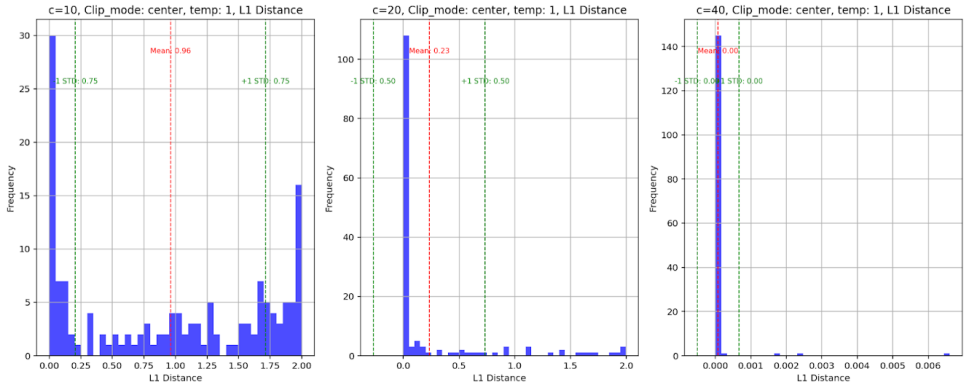
where the supremum is over neighbors $S, S' \in \mathcal{U}$.

Zero-concentrated differential privacy (zCDP) is a relaxation of ε-differential privacy.

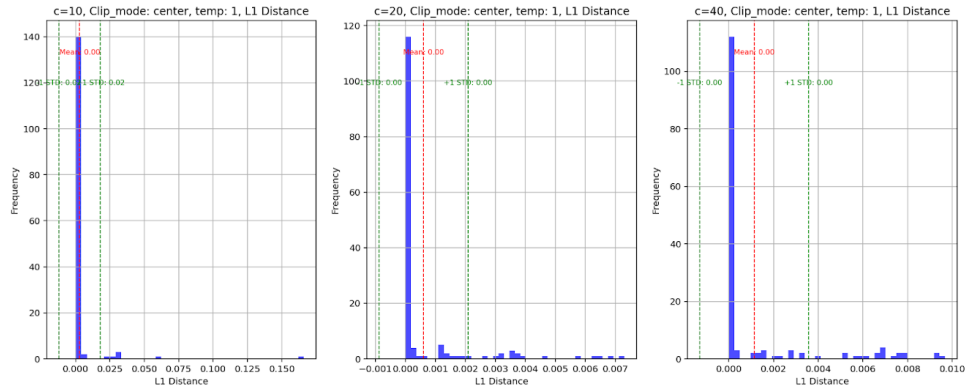
Definition 4 (Bun and Steinke (2016))**.** *A mechanism M satisfies ρ-*zCDP *if*

$$D_\alpha(M(D) \parallel M(D')) \leq \rho\alpha$$

for all $\alpha > 1$ and neighboring datasets $D, D' \in \mathcal{D}$, where $D_\alpha(P \parallel Q)$ is Rényi divergence of order α betweeen distributions P and Q.



(a) Distribution of L1 error induced by fixed interval clipping.



(b) Distribution of L1 error induced by clipping with recentering.

Figure 3: We sample a few hundred tokens using logits aggregation with no clipping. At each sampling step, we compute the L1 distances between the post-softmax distributions of aggregated clipped logits vs. aggregated unclipped logits, at various settings of c, and plot them in a histrogram. We observe less error, at lower choices of c when clipping with recentering (note the x-axis scales).

C.2 Composition

Zero-concentrated differential privacy obeys a simple sequential composition rule.

Lemma 1. *If mechanisms M_1 and M_2 satisfy ρ_1-zCDP and ρ_2-zCDP, respectively, then the sequential composition of M_1 and M_2 satisfies $(\rho_1 + \rho_2)$-zCDP.*

Parallel composition is a well-known technique in differential privacy that is useful for establishing privacy guarantees in scenarios where a mechanism is independently applied to disjoint subsets of a dataset. Many versions of parallel composition require that the subsets are chosen in a fully data-independent manner. We show that the same result holds under a weaker assumption.

Lemma 2. *Let k be a positive integer. Let f be a function that maps prompts into $[k]$. For any dataset of prompts D and $i \in [k]$ let*

$$D_i = \{\mathbf{p} \in D : f(\mathbf{p}) = i\}.$$

Let M be a mechanism that satisfies ρ-zCDP. If $\widehat{M}$ is the mechanism defined by

$$\widehat{M}(D) = (M(D_1), \ldots, M(D_k))$$

then $\widehat{M}$ satisfies ρ-zCDP.

Proof. Let $D, D' \in \mathcal{D}$ be neighboring datasets. Without loss of generality assume $D = D' \cup \{\mathbf{p}\}$, where $\mathbf{p}$ is a prompt. There exists $j \in [k]$ such that $D_i = D'_i$ for all $i \neq j$ and $D_j = D'_j \cup \{\mathbf{p}\}$. We have for all $\alpha > 1$

$$\begin{aligned} D_\alpha(\widehat{M}(D) \| \widehat{M}(D')) &= \sum_{i=1}^{k} D_\alpha(M(D_i) \| M(D'_i)) \\ &= D_\alpha(M(D_j) \| M(D'_j)) \\ &\leq \rho\alpha \end{aligned}$$

□

C.3 Sensitivity analysis

In this we compute the sensitivity of several functions used in Algorithm 1. Each function depends

on a set of logit vectors. Recall that a logit vector is an element of $\mathbb{R}^v$. Let

$$\ell(Z) = \frac{1}{s} \sum_{\mathbf{z} \in Z} \mathrm{clip}_c(\mathbf{z})$$

where $\mathrm{clip}_c(\cdot)$ was defined in Eq. (1). Also recall the distance function defined in Eq. (2):

$$\mathrm{d}(Z, \mathbf{z}) = \left\| \frac{1}{s} \sum_{\mathbf{z}' \in Z} p_{\mathbf{z}'} - p_{\mathbf{z}} \right\|_1$$

where $p_{\mathbf{z}} = \mathrm{softmax}(\mathbf{z})$.

Lemma 3. *The function ℓ has sensitivity $\frac{c}{s}$, and for all $\mathbf{z} \in \mathbb{R}^v$, the function $\mathrm{d}(\cdot, \mathbf{z})$ has sensitivity $\frac{1}{s}$.*

Proof. Let $Z, Z' \subseteq \mathbb{R}^v$ be neighbors. Let $\tilde{\mathbf{z}} \in \mathbb{R}^v$ be the logit vector they do not have in common. We have

$$\left\| \ell(Z) - \ell(Z') \right\|_\infty = \frac{1}{s} \left\| \mathrm{clip}_c(\tilde{\mathbf{z}}) \right\|_\infty \leq \frac{c}{s}.$$

We also have

$$\begin{aligned}
&\left| \mathrm{d}(Z, \mathbf{z}) - d(D', \mathbf{z}) \right| \\
=& \left| \left\| \frac{1}{s} \sum_{\mathbf{z}' \in Z} p_{\mathbf{z}'} - p_{\mathbf{z}} \right\|_1 - \left\| \frac{1}{s} \sum_{\mathbf{z}' \in Z'} p_{\mathbf{z}'} - p_{\mathbf{z}} \right\|_1 \right| \\
\leq& \left\| \frac{1}{s} \sum_{\mathbf{z}' \in Z} p_{\mathbf{z}'} - \frac{1}{s} \sum_{\mathbf{z}' \in Z'} p_{\mathbf{z}'} \right\|_1 \\
=& \left\| \frac{1}{s} \mathbf{p}_{\tilde{\mathbf{z}}} \right\|_1 \\
=& \frac{1}{s}
\end{aligned}$$

where we used the reverse triangle inequality. □

C.4 Constituent mechanisms

In this section we prove privacy guarantees for several simpler mechanisms that we will later compose together to show that Algorithm 1 is private.

Both Algorithms 2 and 3 accept a sensitive prompt dataset and a token sequence as input. Algorithm 2 appends a private token to the token sequence, while Algorithm 3 appends zero or more public tokens to the token sequence. The operation of both algorithms is governed by the parameters of Algorithm 1 (*e.g.*, temperature, noise level, *etc*).

Lemma 4. *Let $A(D, \mathbf{x}_0)$ be Algorithm 2. For each $\mathbf{x}_0 \in \mathcal{X}^*$ the mechanism $M : D \mapsto A(D, \mathbf{x}_0)$ satisfies ρ-zCDP, where $\rho = \frac{1}{2}(\frac{c}{s\tau})^2$.*

Algorithm 2 Private token generation

Input: Sensitive prompt dataset D, initial token sequence $\mathbf{x}_0$

Output: Token sequence $\mathbf{x} \in \mathcal{X}^*$

$\mathbf{x} \leftarrow \mathbf{x}_0$
$Z \leftarrow \{\mathrm{logits}(\mathbf{px}) : \mathbf{p} \in D\}$
$\bar{\mathbf{z}} \leftarrow \ell(Z)$
$x \sim \mathrm{softmax}(\bar{\mathbf{z}}/\tau)$
Append x to $\mathbf{x}$
return $\mathbf{x}$.

Proof. Consider a function $f : \mathcal{D} \to \mathbb{R}^v$ with sensitivity Δ. By an analysis of the exponential mechanism due to Cesar and Rogers (2021),[6] choosing a token according to the distribution $\mathrm{softmax}(\frac{\varepsilon}{2\Delta})$ satisfies $\frac{1}{8}\varepsilon^2$-zCDP. Observe that mechanism M is the exponential mechanism with $f = \frac{1}{\tau}\ell$, which by Lemma 3 has sensitivity $\frac{c}{s\tau}$. □

Algorithm 3 Public token generation

Input: Sensitive prompt dataset D, initial token sequence $\mathbf{x}_0$

Output: Token sequence $\mathbf{x} \in \mathcal{X}^*$

$\mathbf{x} \leftarrow \mathbf{x}_0$
$\hat{\theta} \leftarrow \theta + \mathrm{Laplace}(\sigma)$
while True **do**
  $Z \leftarrow \{\mathrm{logits}(\mathbf{px}) : \mathbf{p} \in D\}$
  $\mathbf{z}_{\mathrm{public}} \leftarrow \mathrm{logits}(\mathbf{p}_{\mathrm{public}}\mathbf{x})$
  $\hat{d} \leftarrow \mathrm{d}(Z, \mathbf{z}_{\mathrm{public}}) + \mathrm{Laplace}(2\sigma)$
  if $\hat{d} \geq \hat{\theta}$ **then**
    Break
  else
    $x \sim \mathrm{softmax}(\mathbf{z}_{\mathrm{public}}/\tau_{\mathrm{public}})$
    Append x to $\mathbf{x}$
return $\mathbf{x}$.

Lemma 5. *Let $A(D, \mathbf{x}_0)$ be Algorithm 3. For each $\mathbf{x}_0 \in \mathcal{X}^*$ the mechanism $M : D \mapsto A(D, \mathbf{x}_0)$ satisfies ρ-zCDP, where $\rho = \frac{2}{(s\sigma)^2}$.*

Proof. Observe that mechanism M is an instance of the AboveThrehold mechanism (Dwork et al., 2009), which accepts a private dataset, a threshold, and a sequence of queries as input. In each iteration, the AboveThreshold mechanism applies the

[6] See also Rogers and Steinke (2021).

next query in the sequence to the dataset and compares it to a noisy threshold, and returns the index of the first query that exceeds the threshold. The queries can be chosen adaptively and adversarially. In mechanism M, each query is specified by a token sequence $\mathbf{x}$, and the index of the first query that exceeds the threshold is determined by the length of the returned token sequence. Furthermore, by Lemma 3 each query has sensitivity $\frac{1}{s}$. Thus by the analysis due to Dwork et al. (2009), mechanism M satisfies $\frac{2}{s\sigma}$-differential privacy, which by Bun and Steinke (2016) implies that mechanism M satisfies $\frac{2}{(s\sigma)^2}$-zCDP. □

C.5 Putting it all together

Consider a sequence of iterations of the inner loop of Algorithm 1 in which the value of t (the private token counter) is constant. Observe that the operation of Algorithm 1 during these iterations is equivalent to the sequential composition of Algorithms 2 and 3, since these iterations generate zero or more public tokens followed by a private token.[7] Moreover, there are at most r such sequences of iterations, since r is an upper bound on the private token counter for any batch. By Lemmas 1, 4 and 5 we have that Algorithm 1 applied to a single batch satisfies ρ-zCDP (where ρ is specified in the statement of Theorem 1). And therefore by Assumption 1 and Lemma 2 we have that Algorithm 1 applied to the whole dataset satisfies ρ-zCDP. It remains to convert this zCDP guarantee to an (ε, δ)-differential privacy guarantee, which we do two different ways using two existing results: Corollary 13 due to Canonne et al. (2020) and Lemma 3.5 due to Bun and Steinke (2016).

D Privacy-equivalent Gaussian noise

Given the average token distribution $\bar{\mathbf{p}}$ in a batch, Tang et al. (2024) protect the privacy of $\bar{\mathbf{p}}$ by using the Gaussian mechanism, which achieves (ε, δ)-differential privacy with $\varepsilon = \frac{\sqrt{2\log(1.25/\delta)}}{s\sigma}$, where s is the batch size and σ is the standard deviation of the noise added to each probability in $\bar{\mathbf{p}}$. On the other hand, we use the exponential mechanism to protect the privacy of a sample drawn from $\bar{\mathbf{p}}$, which achieves ε-differential privacy with $\varepsilon = \frac{2c}{s\tau}$, where c is the maximum absolute value of any log-probability in the batch and τ is the sampling temperature.

Empirically, we obtained good synthetic data quality with $s = 250$, $\tau = 2$, $c = 10$ and $\delta = 10^{-6}$.

Setting the ε values equal to each other yields $\sigma = \frac{\tau\sqrt{\log(1.25/\delta)}}{\sqrt{2}c}$, which is the noise level needed for the two mechanisms to have comparable privacy guarantees (setting aside that $\delta > 0$, an omission that only favors the Gaussian mechanism). Plugging in the above parameters yields $\sigma \approx 0.53$.

The analysis in Theorem 8 of Balle and Wang (2018) does not admit a closed-form solution. Instead, we binary search for a solution to:

$$\Phi\left(\frac{\Delta}{2\sigma} - \frac{\varepsilon\sigma}{\Delta}\right) - \exp(\varepsilon)\Phi\left(-\frac{\Delta}{2\sigma} - \frac{\varepsilon\sigma}{\Delta}\right) \leq \delta$$

where Φ is the Gaussian cdf, $\varepsilon = \frac{2c}{s\tau}$, $\delta = 10^{-6}$, and Δ is the L2 sensitivity of a vector computed as the average of s user-provided probability vectors, namely $\Delta = 1/s$. This procedure yields $\sigma \approx 0.34$.

Finally, equating the zCDP loss for the exponential mechanism given by $\frac{\varepsilon^2}{8} = \frac{c^2}{2s^2\tau^2}$ (Cesar and Rogers (2021)) to that of the Gaussian mechanism given by $\frac{1}{2s^2\sigma^2}$ (Bun and Steinke (2016)), yields $\sigma = 0.2$.

E Experiment details

E.1 Hyperparameter tuning

There are a significant amount of hyperparameters associated with our approach. See Table 7 for a list of the main ones and the values they take. In this section we describe the hyperparameter evaluation procedure, as well as the rationale for our decisions on what hyperparameter settings to couple together or that we altogether avoid running.

Hyperparameter evaluation procedure. For fine-tuning experiments, we set aside a real validation set consisting of 10% the real train set. We choose dataset generation parameters based on which resulting dataset induces the the best classifier on this real validation set. However, the process of tuning the classifier itself on synthetic data (choosing the best learning rate and checkpoint) does not use real data – we do that tuning with synthetic data. This is because the output of our method is a dataset, and its usefulness to train a model includes how well subsets of it can be used for downstream task hyperparameter selection. After identifying the best synthetic dataset in this

[7] The special treatment of the <eos> token complicates this story a little, but we can always assume that the LLM ignores any tokens before the last <eos> token.

manner, we run the tuning process based on synthetic data only and report accuracy of the resultant classifier on the real test set.

Hyperparameter choices. Based on initial experiments, we found that setting $c = 10$ and $\tau = 2$ produced well formed text, so we fix $c = 10$ and try a low temperature ($\tau = 1.5$) and a high temperature ($\tau = 2.25$) setting. At $\tau = 2.25$, we observed text degeneration. This is due to the combination of Gemma's large vocabulary (256K) and clipping, which raises the "probability floor" of nonsense tokens. So for $\tau = 2.25$ settings only, we follow Tang et al. (2024) and reduce the vocabulary to the public prediction's top 1024. We emphasize that (1) we do not do this for any of the other settings of τ, and (2) use a larger value than prior work (they use top 100).

Keeping other parameters fixed and increasing the batch size s decreases ε. At the same time, it raises the amount of compute spent to decode a single example.[8] Hence our approach for selecting the batch size is based on the following: given a target epsilon and dataset, choose s large enough so that we can hit at least 1K examples at the low temperature setting $\tau = 1.5$. When targeting a large ε, choosing large s results in too many tokens to decode at too high of a cost per token.

For the sparse vector hyperparameters, we consider the following paired (θ, σ) settings: $\{(-\infty, -), (0.3, 0.1), (0.5, 0.2), (0.7, 0.3)\}$. The first setting corresponds to no use of the SVT, the next 3 represent different privacy levels per token: moving to the right uses noisier queries (less privacy budget) and more often uses the free public tokens. For large datasets and target ε, we do not run the high privacy settings (too much compute to finish), and for smaller datasets and smaller ε we omit the settings that do not produce at least 1K examples.

E.2 Prompts used

We report the prompts used for our experiments. Generally, we use the same prompt for private and public predictions, with "`<text of xxx>`" in the public prompt replaced with an actual private example in the private prompt. The exception is for WikiMoviesJSON (Figures 11 and 12), where the public prompt contains a schema description in place of the example.

[8] The way we interpret this is that s is a compute multiplier that broadens the search space to include better utility configurations in the low ε regime. This is analagous to the role of the noise multiplier σ in DP-SGD, where the best results at low ε come from taking more steps at higher noise levels.

α	Description	Values
s	batch size	127, 255, 511, 1023, 1535, 2047
c	logits clip bound	10
τ	temperature	1.5, 2, 2.25
θ	SVT threshold	$-\infty$, 0.3, 0.5, 0.7
σ	SVT noise level	$-$, 0.1, 0.2, 0.2
τ_{public}	public temperature	1.5

Table 7: Values for hyperparameters explored in this work.

F Artifacts

Tables 1a and 1b list all artifacts we use in this work. AGNews, TREC, DBPedia, MIT-G, MIT-D, IMDB, and Yelp are all standard academic datasets permissible for research use; we cite their original publications when introduced. WikiMoviesJSON is scraped from Wikipedia data, courtesy of (Rust, 2024); their work is covered by an MIT license. Wikipedia content is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA) and the GNU Free Documentation License (GFDL).

We use open-source models BERT-Base, released by (Turc et al., 2019), and Gemma. Our use of Gemma for academic purposes is in accordance of the Gemma terms of use: `https://ai.google.dev/gemma/terms`. GPT-3 is accessible for academic purposes under OpenAI's terms of use, which supports educational and research activities. LaMDA 8B is not publically available, but we received sufficient authorization to use it for the academic purposes of this paper.

G Compute budget

Our main experiments for synthetic data generation run on Gemma 1.1 2B IT. A run of synthetic data generation takes between 8-48 accelerator hours. Including exploratory runs and hyperparameter search, the total compute budget for this project is roughly 14,000 accelerator hours.

```
# [User]
Here are texts with News Type: Business.

Text: <text of News Type: Business>

Please give me another one.

# [Assistant]
Text:
```

Figure 4: Generation prompt for AGNews.

```
# [User]
Here are questions with Answer Type: Entity.

```
Text: <question of Answer Type: Entity>
```

Please give me another one.

[Assistant]
```
Question:
```

Figure 5: Generation prompt for TREC.

```
# [User]
Here are entries of Category: School.

Entry: <entry of Category: School>

Please give me another one.

# [Assistant]
Entry:
```

Figure 6: Generation prompt for DBPedia.

```
# [User]
Give me text about a film and the extracted Phrase about its Director.

Phrase: "josh trank"
Text: "<text containing phrase "josh trank">"

Please give me another Phrase and Text: "josh trank". IMPORTANT: The exact
Director phrase "josh trank" must be mentioned in Text.

# [Assistant]
Phrase: "josh trank"
Text: "
```

Figure 7: Generation prompt for MIT-D.
```

```
# [User]
Give me text about a film and the extracted Phrase about its Genre.

Phrase "comedy"
Text: "<text containing phrase "comedy">"

Please give me another Phrase and Text. IMPORTANT: The exact Genre phrase
"comedy" must be mentioned in Text.

# [Assistant]
Phrase: "comedy"
Text: "
```

Figure 8: Generation prompt for MIT-G.

```
# [User]
Here are texts with Sentiment: Negative.

Text: <text of Sentiment: Negative>

Please give me another one.

# [Assistant]
Text:
```

Figure 9: Generation prompt for IMDB.

```
# [User]
Here are texts with Sentiment: Negative.

Text: <text of Sentiment: Negative>

Please give me another one.

# [Assistant]
Text:
```

Figure 10: Generation prompt for Yelp.

```
# [User]
Here is a JSON record:
```
{
  "title": "$50,000 Reward",
  "year": 1924,
  "cast": [
    "Ken Maynard",
    "Esther Ralston"
  ],
  "genres": [
    "Western",
    "Silent"
  ],
  "href": "$50,000_Reward",
  "extract": "$50,000 Reward is a 1924 American silent Western film directed
by Clifford S. Elfelt and starring Ken Maynard, Esther Ralston and Bert
Lindley."
}
```
Please give me another JSON record that complies with the above schema.

# [Assistant]
```
{
```
```

Figure 11: Private generation prompt for WikiMoviesJSON.

```
# [User]
Here is the schema for a JSON record:
Schema:
```
{
  "title": "str",
  "year": int,
  "cast": [# list of str
    "str1", # 0 or more total entries
  ],
  "genres": [# list of str
    "str1", # 0 or more total entries
  ]
  "href": "str", # URL slug, e.g.: Link_to_Page
  "extract": "str"
}
```
Please give me another JSON record that complies with the above schema.

# [Assistant]
```
{
```

Figure 12: Public generation prompt for WikiMoviesJSON.
```