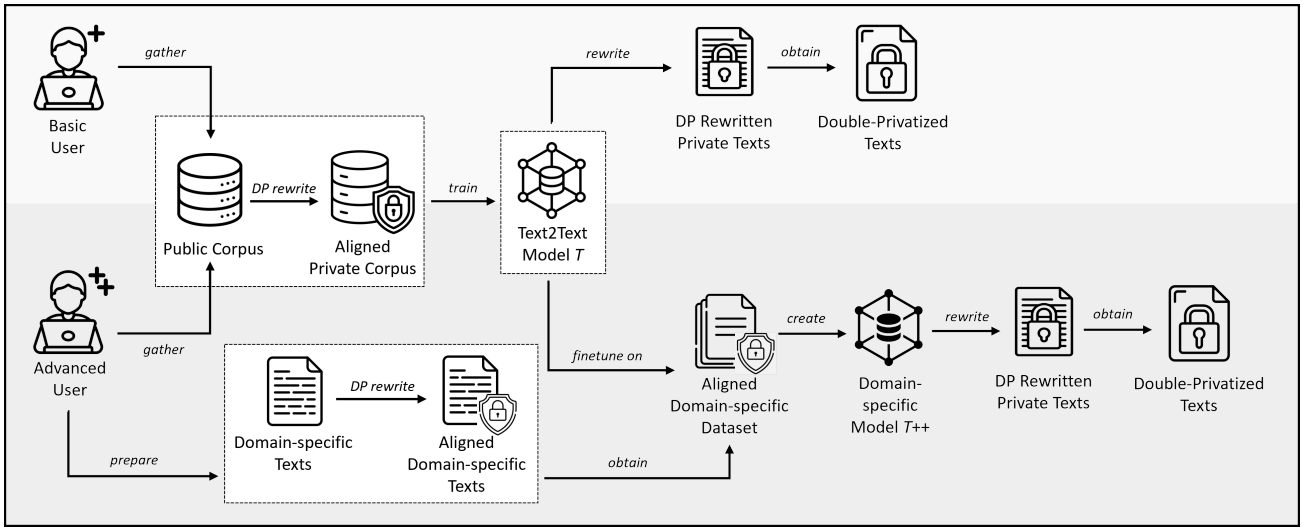

Figure 2: A post-processing pipeline to improve the semantic coherence and privacy preservation of DP rewritten text. Both "tracks" involve a user creating a model T to rewrite DP rewritten text again. While a basic user utilizes large-scale public text corpora, the more advanced user (bottom track) also leverages domain-specific data to fine-tune T further to obtain T++.

The motivation of the user is to re-align DP rewritten output text to be semantically closer to the original meaning of the text, thereby boosting the utility and fidelity of the text eventually to be released. As it turns out, an added benefit of performing this extra step also is the enhanced privatization of the output text, providing an added incentive for the user. These benefits will be empirically demonstrated in Section 5 and analyzed in Section 6.

3.2 Method

The following outlines the basic steps taken by the user in our proposed post-processing pipeline to rewrite the DP rewritten texts once again. The ensuing narrative then goes into detail on each of these steps. The user takes the following steps:

(1) Collect a large number of texts from a public text corpus
(2) Use the rewriting mechanism $\mathcal{M}$ with parameter ε to rewrite the public corpus into a *aligned private corpus*
(3) Finetune a Text2Text LM to generate text in the *opposite* direction, i.e., generate the public corpus text given the aligned corpus text as input
(4) Use the Text2Text LM to "re-rewrite" the DP rewritten texts
(5) Release the doubly rewritten texts

In a more advanced setup, the user takes the same steps (1)-(3), but then continues as follows:

(4) Generate *aligned private texts* from the same domain as the DP rewritten texts
(5) Further finetune the Text2Text LM on this parallel domain-specific data, once again in the "reverse" direction
(6) Use the further finetuned Text2Text LM to "re-rewrite" the DP rewritten texts
(7) Release the doubly rewritten texts

3.2.1 Corpus Creation. The first step of our proposed post-processing pipeline comes with the collection of a large collection of text samples from a publicly available corpus, such as Wikipedia [13] or Common Crawl [31]. Using these text samples, a user can use the DP rewriting mechanism $\mathcal{M}$ to rewrite the samples in DP rewritten versions, thus creating a parallel, *aligned* corpus.

For the purposes of this work, we utilize a random sample of 100,000 English text samples from C4 (Colossal Cleaned Crawled Corpus), made available by Raffel et al. [31] and cleaned by allen.ai[1].

3.2.2 LM Fine-Tuning. Given the prepared aligned corpus, the next task is to fine-tune a Text2Text LM. Much like in the tasks of Machine Translation of Text Simplification, such a fine-tuning task requires a set of parallel *source-target* text pairs, which are naturally provided in the DP rewriting process. With this, the DP rewritten texts become the *source* documents, and the original texts represent the *target* documents. This setup is then used to fine-tune an LM, with the effective aim of "shifting" the rewritten texts back to the general semantic makeup of the original texts.

3.2.3 Track Two: Further Fine-Tuning. As introduced in the above steps, a more advanced user may opt to fine-tune the Text2Text model even further, given the presence of domain-specific texts that more closely mirror the true texts to be released. The motivation behind this is that large-scale public text corpora may contain a vast variety of text subjects, which may not be completely suitable given a set of sensitive texts in one specific domain, that must, in turn, be privatized (rewritten).

With this domain-specific data, the user would proceed as before with the public corpus, i.e., first generate the parallel, aligned dataset, and then proceed to fine-tune the previously obtained Text2Text model further. The output is thus a model that is more

[1] https://huggingface.co/datasets/allenai/c4

tuned to the target domain, with the ability to recapture the semantics in a particular subject area.

A representative example of domain-specific data, one that we leverage and evaluate later in this work, is that of *user reviews*. If a user is to privatize his or her personal reviews before releasing them, publicly available datasets such as Yelp Reviews [41] or Trustpilot Reviews [16] may be useful.

3.2.4 Data Rewriting and Release. The final step, given the (domain-specific) fine-tuned LM, is to use the model to rewrite the DP rewritten text outputs. Only these "doubly" rewritten text outputs are then shared with third parties, and both the original and DP rewritten texts remain private to the user.

4 EXPERIMENTAL SETUP

We describe our experimental setup, which includes the selected DP rewriting mechanisms, design choices for our proposed method, and empirical privacy experiment parameters.

4.1 Selected DP Mechanisms

For the evaluation of our proposed method, we choose two DP rewriting mechanisms from the recent literature.

DP-BART. The DP-BART mechanism [19] was proposed as an LDP mechanism using the BART model architecture [22] as its basis. In essence, DP calibrated noise is added to the latent text representation existing between the encoder and decoder components of BART. As such, the rewriting mechanism operates at the *document-level*, where a single output document is generated in a DP manner given an input document. In particular, we utilize the **DP-BART-CLV** variant as proposed by Igamberdiev and Habernal [19], which clips the latent vector values before adding noise.

For ε values, we choose the first and third quartile values from the range of values evaluated by the original authors. This corresponds to the ε values of *625* and *1875*.

DP-Prompt. Leveraging the proxy task of *paraphrasing*, DP-Prompt [34] presents a method to generate privatized documents under LDP guarantees by using (large) language models as paraphrasers. Utilizing a temperature sampling mechanism as a form of Exponential Mechanism [25], privatized documents are generated word-by-word by a DP sampling of the next token, thus providing DP guarantees at the *word-level*. For our implementation of DP-Prompt, we use the FLAN-T5-large language model (780M parameters) [6].

As with DP-BART, we use two ε values as used by the authors of DP-Prompt, namely *137* and *206*. Specifically, following Utpala et al. [34], we first measure the logit range of our selected Flan-T5-Large model. This is done by estimating the range via running the model on 100 randomly selected texts from the C4 corpus, and accordingly choosing the clipping range to be $(logit_{min}, logit_{max}) = (-95, 8)$, thus leading to a sensitivity of 103. Next, we choose the temperature values T of 1.0 and 1.5 (first and third quartile values), which correspond to the two ε values listed above[2].

[2]Rounded values from the formula $\varepsilon = \frac{2 \cdot \Delta}{T}$, where Δ represents the sensitivity.

4.2 Post-Processing Pipeline

Datasets. As stated in Section 3.2.1, we use a random sample of 100,000 text samples from the C4 Corpus to serve as the *public corpus* as envisioned in our post-processing pipeline. Additionally, we use two *domain-specific* datasets, namely the Yelp and Trustpilot reviews, which will be covered in more detail in the following outline of the empirical privacy experimental setup.

Model Fine-Tuning. In both fine-tuning scenarios, that is the base fine-tuning to create model T and the further fine-tuning to obtain T++, we once again employ the FLAN-T5-large model checkpoint. Given the input parallel corpus (public-private or domain-specific), the model is trained for one epoch with a learning rate of 5e-5, and otherwise all default HuggingFace Trainer[3] parameters. This process resulted in a total of 12 trained models, namely:

Basic User:

(1) Model T, fine-tuned on *aligned private corpus* (*apc*) with DP-BART (ε=625)
(2) Model T, fine-tuned on *apc* with DP-BART (ε=1875)
(3) Model T, fine-tuned on *apc* with DP-Prompt (ε=137)
(4) Model T, fine-tuned on *apc* with DP-Prompt (ε=206)

Advanced User:

(1) Model T++, further fine-tuned on the Yelp *aligned domain-specific corpus* with DP-BART (ε=625)
(2) Model T++, further fine-tuned on the Yelp *aligned domain-specific corpus* with DP-BART (ε=1875)
(3) Model T++, further fine-tuned on the Yelp *aligned domain-specific corpus* with DP-Prompt (ε=137)
(4) Model T++, further fine-tuned on the Yelp *aligned domain-specific corpus* with DP-Prompt (ε=206)
(5) Model T++, further fine-tuned on the Trustpilot *aligned domain-specific corpus* with DP-BART (ε=625)
(6) Model T++, further fine-tuned on the Trustpilot *aligned domain-specific corpus* with DP-BART (ε=1875)
(7) Model T++, further fine-tuned on the Trustpilot *aligned domain-specific corpus* with DP-Prompt (ε=137)
(8) Model T++, further fine-tuned on the Trustpilot *aligned domain-specific corpus* with DP-Prompt (ε=206)

4.3 Empirical Privacy Experiments

With the post-processing pipeline, the goal is now to evaluate its effect on the *empirical privacy* protection provided by DP rewriting mechanisms. Additionally, we also measure the effects of the post-processing step on the *utility* of the text, an important counterbalance to be measured in DP text rewriting [24, 28].

4.3.1 Datasets. As mentioned in Section 4.2, we employ two datasets, both of which allow for direct empirical privacy measurement.

Yelp Reviews. The Yelp Review dataset [41] is a dataset containing user reviews from the popular Yelp platform. Each review contains a star rating from 1-5, leading to the dataset often being used for sentiment analysis. In particular, we use the same data subset as used by Utpala et al. [34], which contains 17,295 reviews from 10 frequent users. Thus, we model the empirical privacy task as an

[3]https://huggingface.co/docs/transformers/en/main_classes/trainer

authorship identification task, wherein an adversary attempts to infer the author given a written text.

Trustpilot Reviews. The Trustpilot Review dataset [16] is a large corpus of reviews from the Trustpilot platform. As with Yelp, each Trustpilot review is scored from 1-5. Additionally, the dataset also lists the gender of each reviewer, thus leading to the *gender identification* adversarial task. As the original dataset is quite expansive, we take a random 10% of reviews with the gender listed, resulting in an evaluation dataset of 29,490 reviews.

4.3.2 Modeling Adversaries. As introduced in Section 2.2, we model two types of adversaries for the empirical privacy evaluation: the *static* and *adaptive* attackers. In order to mimic the data available that these attackers would leverage we perform the following steps:

(1) For each dataset (Yelp/Trustpilot), we take a 90-10 train-validation split, using a random seed of 42.
(2) The train split is known and used by the attacker. The static attacker only has access to the original version of the texts, while the adaptive attacker has access to the privatized (rewritten) versions.
(3) The validation split represents the "true" private text, i.e., the text which the user is releasing, and which the user aims to protect further using post-processing. This text, released in privatized form, is the target of both adversaries.

Given the train split as described above, the attacker in question trains an adversarial model to infer the sensitive attribute, i.e., author or gender. The static attacker trains on the original (clean) texts, while the adaptive attacker trains on the privatized texts (rewritten by mechanism $\mathcal{M}$).

To build these adversarial classifiers, a DeBERTa-v3-base model [14, 15] is fine-tuned. The model is trained with a 10-class classification head for the authorship identification task, and a 2-class head for gender identification. Training is run for one epoch on a given dataset with a learning rate of 5e-5 and otherwise default training parameters. With the static attacker, a single model is trained and then evaluated on all DP rewritten variants, while for the adaptive attacker, a model is trained for each variant and subsequently evaluated on the corresponding privatized validation set. All model training is performed on a single NVIDIA RTX6000 48GB GPU.

4.4 Metrics

Empirical privacy is measured by the ability of a mechanism's rewritten text to *reduce* the adversarial advantage of a given attacker. In this study, this is represented by the reduction in F1 score of the attacker evaluating an adversarial model on the private validation split. This reduction is presented against the baseline of training *and* testing on the clean, non-privatized version of the data, i.e., what an attacker could achieve given full access to the entire target dataset. As such, a lower F1 score represents a higher empirical privacy result. It is important to note that all scores represent an average of three runs, that is, a model is trained three times on different shuffles of the dataset, and the three evaluation results are averaged for the final score.

We also measure the preserved *utility* of the (doubly) rewritten text data, modeled as the *semantic similarity* between the (*original*, *rewritten*) and (*original*, *doubly rewritten*) pairs of text. To measure semantic similarity, we utilize Sentence Transformer models [32], specifically the all-mpnet-base-v2 and all-MiniLM-L6-v2 encoder models. The cosine similarity between the encoded pairs of texts is taken and subsequently averaged over an entire rewritten dataset. Since two models are used to account for model-specific differences, the cosine similarity (CS) scores reported represent the average score between the two models.

5 EMPIRICAL RESULTS

The results of our empirical privacy experiments, namely on the Yelp and Trustpilot datasets, are given in Table 1.

As noted in Table 1, we present the empirical privacy results in the form of *adversarial advantage*, or specifically the F1 score achieved by an adversary (static or adaptive) in all selected scenarios. The baseline scenario depicts the performance an attacker could achieve given full access to the original, non-privatized data. Then, given a (*mechanism*, ε) pair, we present the empirical privacy results for each rewriting scenario: (1) base output from the DP rewriting mechanism, (2) double-privatized outputs from our "basic user", and (3) double-privatized outputs from our "advanced user".

The results also include the *CS* score, as introduced previously, which captures the degree to which semantic meaning is kept from the original text to the rewritten text. Thus, we pose that a higher *CS* score indicates a closer preservation of original semantic meaning.

6 DISCUSSION

We now analyze in-depth the empirical results presented in Table 1, as well as discuss the merits and limitations of our method.

6.1 The Benefits of Post-Processing

An analysis of the empirical results reveals the strengths of our post-processing method, particularly in reducing adversarial performance. Concretely, in the 16 attacker scenarios presented (8x static, 8x adaptive), either our *basic* or *advanced* method leads to the lowest adversarial performance in 13 of the 16 scenarios. In some cases, particularly with DP-BART, the adaptive attacker's performance is nearly 50% lower as compared to the DP rewritten texts without post-processing. Similarly, our method also proves to be quite useful in reducing the performance of the static attacker.

Looking to both the static and adaptive scores, one can note that only the texts resulting from our methods can truly *neutralize* the adversarial advantage. In other words, these attackers achieved scores equal to or worse than *majority-class guessing*, i.e., simply choosing one class known to be the most frequent. In the Yelp dataset, the most frequent author writes 17.5% of the reviews, and in Trustpilot, the split is 57.9%/42.1% for male/female. As can be seen in Table 1, only results from the *Basic* or *Advanced* rewritten texts even lead to F1 scores lower than these majority class percentages[4].

A strength of our method comes with the *advanced* setting, which shows in some cases *both* strong protection against adversaries *and* higher semantic similarity to the original texts than only once-rewritten texts. If one assumes cosine similarity (*CS*) to be an indicator of preserved utility, this becomes a quite interesting finding, in contrast to the belief that higher privacy always comes at the

[4]As majority class guessing will result in zero false negatives but many false positives.