BRR: Preserving Privacy of Text Data Efficiently on Device

Ricardo Silva Carvalho [1] **Theodore Vasiloudis** [2] **Oluwaseyi Feyisetan** [2]

Abstract

With the use of personal devices connected to the Internet for tasks such as searches and shopping becoming ubiquitous, ensuring the privacy of the users of such services has become a requirement in order to build and maintain customer trust. While text privatization methods exist, they require the existence of a trusted party that collects user data before applying a privatization method to preserve users' privacy. In this work we propose an efficient mechanism to provide metric differential privacy for text data on-device. With our solution, sensitive data never leaves the device and service providers only have access to privatized data to train models on and analyze. We compare our algorithm to the state-of-the-art for text privatization, showing similar or better utility for the same privacy guarantees, while reducing the storage costs by orders of magnitude, enabling on-device text privatization.

1. Introduction

As users interact with more and more edge devices through text and speech, organizations are able to train models on the data received from users to offer better services and drive business. However, research has demonstrated that machine learning models trained on sensitive data can be attacked, and sensitive information about individuals extracted from the original training data(Shokri et al., 2017).

In order to build and maintain customer trust, companies must protect the privacy of their users, so that they will continue to use and be delighted by the products offered, without mistrust towards the platform and the data holder. The best way to ensure that is by sending no sensitive data to the service provider at all, known as the *zero-trust* model(Dwork et al., 2006). That way sensitive data never leaves the device, and users can be assured that even in the event of a data breach of the service provider, their private data will not be compromised.

Differential Privacy (DP) (Dwork et al., 2006) has emerged as the most well-established technique to provide privacy guarantees for individuals. However, DP was originally designed to deal with continuous data, while categorical and text data pose an additional challenge (Dwork and Roth, 2014). DP works by applying noise to inputs, and for a naive approach to work for text in the *zero-trust* model, it would have to ensure that a word can be replaced by *any* other word in the vocabulary, potentially ruining the utility of the algorithm.

To deal with this restriction, previous research has proposed applying a privatization step in the continuous embedding space instead (Fernandes et al., 2018), using a generalization of DP for metric spaces called *metric-DP* (Chatzikokolakis et al., 2013). These methods work by retrieving the embedding vector of the word we want to privatize, adding noise, then replacing the word with the one closest to the new noisy vector in the embedding space. However, these methods assume a central authority that gathers the data of all the users and applies the privatization mechanism to generate a new privatized dataset, before training downstream NLP models. If we try using these mechanisms on-device, the space cost would be prohibitive. The embedding vectors plus the nearest neighbor index necessary to retrieve the perturbed words from the noisy embedding can reach several gigabytes.

In this paper we propose an approach that allows for zero-trust text data privatization, ensuring that sensitive data never leave the device of the user, improving trust to the service. The key component of our approach is to use binary embedding vectors (Tissier et al., 2019) to dramatically shrink the space and computational costs of storing and querying the word embeddings, while maintaining semantic meaning. Using binary word embedding vector representations, we propose a text privatization mechanism based on the randomized response (Wang et al., 2016), and prove that the mechanism satisfies metric-DP.

In summary, our contributions are the following:

- We propose a zero-trust algorithm for on-device text privatization, using binary embeddings and random-

[1]Simon Fraser University [2]Amazon.com. Correspondence to: Ricardo Silva Carvalho <rsilvaca@sfu.ca>, Theodore Vasiloudis <thvasilo@amazon.com>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

ized response.

- A proof that our mechanism satisfies metric-DP is included, more specifically for Hamming distance.
- We develop theoretical methods for comparing metric-DP mechanisms that use different metrics, allowing consistent privacy-utility evaluation.
- Finally, our empirical evaluation demonstrates the computational advantages of the approach compared to the state-of-the-art, while maintaining better or similar utility.

2. Related Work

Fernandes et al. (2018) were one of the first to use metric differential privacy on text data. They focused on the "bag of words" representation of documents and applied the Earth Mover's metric to obtain privatized bags, being also the first work to perform individual word privatization in the context of metric differential privacy. Following this context, the Madlib[1] mechanism (Feyisetan et al., 2020) adds noise to embedding vectors of words, working on the Euclidean space and adding Laplacian noise. After introducing noise, the mechanism outputs the word that is closest to the noisy vector in the embedding space. The algorithm presented in (Feyisetan et al., 2019) is a follow-up to (Feyisetan et al., 2020) although it appeared later. This mechanism works in a hierarchical embedding space, where the embedding vector of an input word is perturbed with noise from a hyperbolic distribution. These works successfully illustrated the privacy-utility trade-off on metric differential privacy, and empirically showed that we can achieve reasonable privacy guarantees with the impact on the utility of downstream text models being dependent on the complexity of the downstream task. For example, the complex question-answering task was more affected than binary classification.

Feyisetan et al. (2019) compare the hyperbolic mechanism to Madlib (Feyisetan et al., 2020). However, since the two algorithms use different metric functions, the evaluation of privacy via only matching the ε parameter of differential privacy can be improved. In this sense, Feyisetan et al. (2019) compares the privacy of the two mechanisms, looking at the probability of not changing a word after noise injection, i.e. the probability that the mechanism returns the exact same word used as input. Even though this notion can be intuitively seen as a level of indistinguishability, it cannot guarantee a fair comparison between mechanisms. In Section 5 we propose a method that ensures a more fair comparison, based on a privacy loss bound.

[1]We refer to this algorithm with the name used in (Diethe, 2020)

Finally, to the best of our knowledge, our work is the first to apply metric differential privacy on text data efficiently on-device. This application scenario allows users to share only already privatized data, keeping their sensitive information local. In this sense, our intended use is similar to the goals of Local Differential Privacy (LDP)(Dwork et al., 2006). Nonetheless, previous work on LDP focused on aggregate statistics, instead of individual word privatization. Examples are Google's RAPPOR (Fanti et al., 2016), Apple's DP distributed system (Team, 2017) and Microsoft's Private Collection of Telemetry Data (Ding et al., 2017). In the context of individual privatization of a given input, in our case word, using LDP would mean adding extremely amounts of noise, which metric-DP relaxes by including distance metrics within the differential privacy guarantees.

3. Preliminaries

Consider a user giving as input a word w from a discrete fixed domain $\mathcal{W}$. For any pair of inputs w and w', we assume a distance function $d : \mathcal{W} \times \mathcal{W} \to \mathbb{R}_+$, in a given space of representation of these words. Specifically, we consider a word embedding model $\phi : \mathcal{W} \to \mathbb{R}^n$ will be used to represent words, and the distance function can be a valid metric applicable to the embedding vectors.

Our goal is to select a word from $\mathcal{W}$, based on a given input, such that the privacy of the user, with respect to their word choice, is preserved. From an attacker's perspective, the output of an algorithm working over an input w or w' will become more similar as these inputs become closer according to the distance $d(w, w')$. In other words, if two words are close on the embedding space, they tend to generate similar results with same probabilities.

With that in mind, we will work on Metric-Differential Privacy (Chatzikokolakis et al., 2013), a privacy standard defined for randomized algorithms with input from a domain $\mathcal{W}$ that are equipped with a distance metric $d : \mathcal{W} \times \mathcal{W} \to \mathbb{R}_+$ satisfying the formal axioms of a metric. In this context, algorithms satisfying metric-DP will have privacy guarantees that depend not only on the privacy parameter ε, but also on the particular distance metric d being used.

Definition 3.1. *(Metric Differential privacy (Chatzikokolakis et al., 2013)) Given a distance metric $d : \mathcal{W} \times \mathcal{W} \to \mathbb{R}_+$, a randomized mechanism $\mathcal{M} : \mathcal{W} \to \mathcal{Y}$ is εd-differentially private if for any $w, w' \in \mathcal{W}$ and all outputs $y \in \mathcal{Y}$ we have:*

$$\Pr[\mathcal{M}(w) = y] \leq e^{\varepsilon d(w,w')} \Pr[\mathcal{M}(w') = y] \qquad (1)$$

Usually, on the standard definition of differential privacy(Dwork et al., 2006), the privacy guarantees provided by different mechanisms are compared by looking at the ε value, such that mechanisms with same ε give the same pri-

Algorithm 1 - **Madlib**: Word Privatization Mechanism for Metric Differential Privacy

input Finite domain $\mathcal{W}$, input word $w \in \mathcal{W}$ and privacy parameter ε.
output Privatized word $\hat{w}$.
1: Compute embedding $\phi_w = \phi(w)$
2: Perturb embedding to obtain $\hat{\phi}_w = \phi_w + N$ with noise density $p_N(z) \propto \exp(-\varepsilon \|z\|)$
3: Obtain perturbed element: $\hat{w} = argmin_{y \in \mathcal{W}} \left\| \phi(y) - \hat{\phi}_w \right\|$
4: Return $\hat{w}$

vacy guarantee. For a fair evaluation on metric-DP using ε, we also have to consider the distance metrics used, since it also affects the privacy guarantees, as we can see from Definition 3.1. We describe our theoretically motivated method to enable a fair privacy comparison of mechanisms with different metrics in Section 5.

For Euclidean distance as metric, as discussed on Section 2, the current state-of-the-art is the Madlib mechanism. It uses the Laplace mechanism to add Laplacian noise to a given vector in order to obtain a private output.

For a Euclidean metric $d : \mathcal{W} \times \mathcal{W} \rightarrow \mathbb{R}_+$, Madlib provides metric differential privacy.

Theorem 1. *For a Euclidean distance metric d, Algorithm 1 is εd-differentially private. Proof in (Feyisetan et al., 2020).*

Next we develop our algorithm that satisfies metric-DP, giving formal proof of its privacy guarantees.

4. Mechanism

Our proposed mechanism will employ metric differential privacy to sanitize words via their embedding vectors. The challenge with protecting the privacy is that, for any given input word, the output of a DP algorithm as defined in (Dwork et al., 2006) can be any word in the vocabulary, i.e. the outputs of a mechanism for any pair of inputs words are relatively similar. Metric-DP provides a generalization of DP that allows adjusting the privacy of an input by leveraging a given distance metric, thus being a suitable framework for improved utility in the text scenario.

However, algorithms like Madlib rely on having access to word embedding vectors and an approximate nearest neighbor index to map noisy vectors to words. The space cost of these can range from hundreds of MB to several GBs. Thus, using such representations to sanitize words on a user's device would be impractical. In this context, recent research (Tissier et al., 2019; Shen et al., 2019) has focused on converting pre-trained real valued embedding vectors into binary representations. They explicitly aim at keeping the semantic meaning while transforming representations. These works show experimental results on machine learning tasks with the binarization of word embeddings leading to a loss of approximately 2% in accuracy, with the upside of reducing the embedding's size by 97%. Thus, in this work we propose to use binary embeddings of words, obtained from transforming publicly available continuous representations, such as GloVe (Pennington et al., 2014) or FastText (Bojanowski et al., 2017). In addition, specialized nearest neighbor indexes for binary vectors exist (Norouzi et al., 2012), that dramatically reduce the space and time cost of nearest neighbor retrieval.

Randomized Response (RR) (Warner, 1965) is a mechanism that provably (Wang et al., 2016) obtains better utility than the classical Laplace mechanism for binary data collection. RR is a method that dates back to 1965, its original purpose being to motivate survey respondents to answer questions truthfully, without the risk of exposing any private information. For sensitive yes/no questions, survey participants would use a spinner, similar to flipping a coin, and based on its outcome, would either respond truthfully if, for example, the coin came up heads, or respond yes otherwise. This mechanism provides individuals plausible deniability, while allowing researchers to de-bias the results and obtain the aggregate metrics they need. In our case, RR flips a given input bit of an embedding vector with probability inversely proportional to the privacy parameter ε. We describe a general version (Wang et al., 2016) of RR on Algorithm 2.

Algorithm 2 - **RR**: Randomized Response

input Bit $b \in \{0, 1\}$ and privacy parameter ε.
output Privatized bit $\hat{b}$.
1: Set $\hat{b} = b$ with probability $\frac{e^\varepsilon}{1+e^\varepsilon}$, otherwise $\hat{b} = 1 - b$
2: Return $\hat{b}$

RR satisfies metric differential privacy with respect to the privacy parameter ε and Hamming distance. Since this is the first time RR is applied to metric-DP, we include a proof of its privacy guarantees.

Theorem 2. *For a Hamming metric d, Algorithm 2 is εd-differentially private.*

Proof. For RR to satisfy metric differential privacy with Hamming distance metric d, we have to show that, for two bits b and b' and response bit y we get:

$$\frac{\Pr[RR(b) = y]}{\Pr[RR(b') = y]} \leq e^{\varepsilon d(b,b')} \tag{2}$$

For $b = b'$ we have $\Pr[RR(b) = y] = \Pr[RR(b') = y]$ and also $d(b, b') = 0$, therefore Equation 2 is satisfied.

For $b \neq b'$, from the definition of RR, we have:

$$\frac{\Pr[RR(b) = y]}{\Pr[RR(b') = y]} \leq \frac{\frac{e^{\varepsilon}}{1+e^{\varepsilon}}}{\frac{1}{1+e^{\varepsilon}}} = e^{\varepsilon}$$

Since for $b \neq b'$ we also get $d(b, b') = 1$, this means we also have Equation 2 satisfied for this case.

□

We now describe our mechanism, denoted as Binary embeddings over Randomized Response (BRR), described in Algorithm 3. BRR uses binary embedding vectors to represent words and applies RR to make each binary vector differentially private.

Algorithm 3 - **BRR**: Mechanism for Text as Binary Embeddings over Randomized Response

input Finite domain $\mathcal{W}$, input word $w \in \mathcal{W}$ and privacy parameter ε.

output Privatized word $\hat{w}$.

1: Compute **binary** embedding vector $\phi_w = \phi(w)$
2: Perturb word embedding vector using **Randomized Response** to obtain $\hat{\phi}_w = RR(\phi_w, \varepsilon)$
3: Obtain perturbed word: $\hat{w} = argmin_{y \in \mathcal{W}} \left\| \phi(y) - \hat{\phi}_w \right\|$
4: Return $\hat{w}$

With the algorithm described, we state the privacy guarantees of BRR.

Theorem 3. *For a Hamming metric d, Algorithm 3 is εd-differentially private.*

Proof. For embeddings that are independent of the data, the nearest neighbor search is just a post-processing step, thus without privacy loss. Therefore we only have to analyze the privacy of releasing the perturbed embedding vector.

Consider d as the Hamming distance and M as the mechanism inside BRR that performs the embedding perturbation, such that each word $w \in \mathcal{W}$ has a binary embedding representation $\phi_w : \{0,1\}^n$. Then for a given output vector $y : \{0,1\}^n$, with i'th bit as y_i and any pair of inputs $w, w' \in \mathcal{W}$, with i'th bits as w_i and w'_i, from using RR on a single bit position i we have that:

$$\frac{\Pr[M(w_i) = y_i]}{\Pr[M(w'_i) = y_i]} \leq e^{\varepsilon d(w_i, w'_i)}$$

For n bits, multiplying probabilities for each of them, we then have:

$$\frac{\Pr[M(w) = y]}{\Pr[M(w') = y]} = \prod_{i=1}^{n} \frac{\Pr[M(w_i) = y_i]}{\Pr[M(w'_i) = y_i]} \leq e^{\varepsilon \sum_{i=1}^{n} d(w_i, w'_i)} = e^{\varepsilon d(w, w')}$$

where the last step comes from the definition of the Hamming distance.

□

BRR is similar to the Madlib Mechanism, differing in the use of binary embeddings instead of real-valued, and Randomized Response instead of the Laplace mechanism. Since RR is proven to be better than Laplace mechanism for binary data (Wang et al., 2016), if our semantic loss for transforming embeddings into binary is smaller than the gains of using RR instead of Laplace, then BRR is a promising approach compared to Madlib.

More importantly, our mechanism is suitable to privatize data on-device due to the use of binary embeddings and specialized nearest neighbor index. On one side, we have the memory/storage size reduction of using binary embedding vectors, and additionally we gain computational efficiency in the perturbation with RR, implemented as sampling from a binomial distribution together with a XOR operation, which can be done efficiently at the hardware level. With these optimizations, the user would only need to share already privatized data, keeping ownership of the sensitive data. This is a highly desirable feature, as it allows the use of valuable user data for NLP tasks, but sensitive information never leaves the user's device.

5. Comparing Metric-DP Mechanisms

One issue with a fair evaluation of BRR against Madlib is that they use different distance functions. To solve this we propose fixing a *privacy ratio* that allows us to obtain similar privacy guarantees, even when two mechanisms use different distance metrics. Due to space limitations we provide a brief description here and refer the interested reader to Appendix A for a detailed motivation of the method.

To compare mechanisms with different distance metrics, we consider an estimate of *privacy loss bound* $\varepsilon \cdot \mathcal{P}_d$, where $\mathcal{P}_d$ is defined as an aggregate distance measurement based on the distances between all possible pairs of words. In this work, we use either $\mathcal{P}_d^{avg}$ where we average the distances of all pairs, or $\mathcal{P}_d^{max}$ where we use the maximum distance between any two words, for each mechanism. To fairly compare the privacy of two mechanisms, we equalize their bounds via a privacy ratio.

Definition 5.1 (Method to Fix Privacy)**.** *Given two randomized mechanisms $\mathcal{M}_A$ and $\mathcal{M}_B$ both taking inputs from $\mathcal{X}$ to*

output space $\mathcal{Y}$, satisfying respectively $\varepsilon_A d_A$-DP and $\varepsilon_B d_B$-DP, we denote the privacy ratio as $\mathcal{R}_{d_A,d_B} = \mathcal{P}_{d_A}/\mathcal{P}_{d_B}$. In order to ensure a similar privacy loss on both mechanisms, for any given ε_A defined for $\mathcal{M}_A$ we need to set:

$$\varepsilon_B = \mathcal{R}_{d_A,d_B} \cdot \varepsilon_A \tag{3}$$

In practice, to compare two mechanisms $\mathcal{M}_A$ and $\mathcal{M}_B$, first we calculate the aggregate distances, e.g. max or average distances between all words, $\mathcal{P}_{d_A}$ and $\mathcal{P}_{d_B}$. Then we obtain the privacy ratio $\mathcal{R}_{d_A,d_B} = \mathcal{P}_{d_A}/\mathcal{P}_{d_B}$. Then for experiments we can simply choose for $\mathcal{M}_A$ any value of ε_A and then for $\mathcal{M}_B$ we set $\varepsilon_B = \mathcal{R}_{d_A,d_B} \cdot \varepsilon_A$. This implies that $\varepsilon_B \cdot \mathcal{P}_{d_B} = \varepsilon_A \cdot \mathcal{P}_{d_A}$, fixing the estimate of privacy loss for the two mechanisms considered.

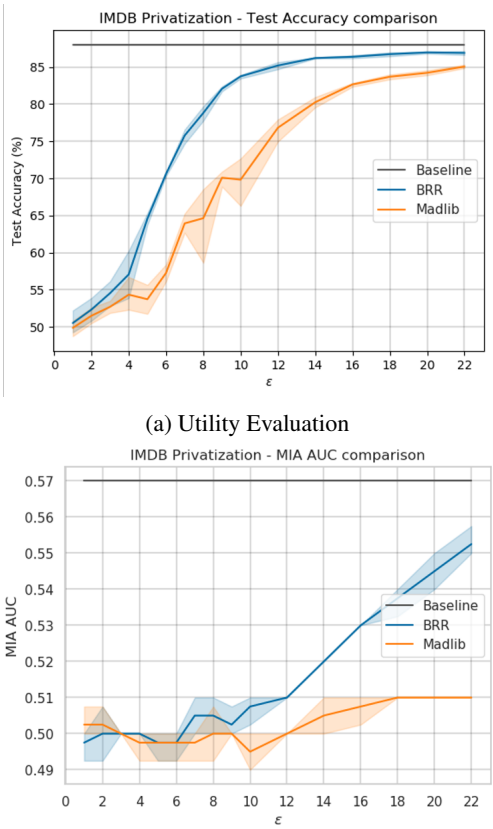
By calculating the privacy ratio $\mathcal{R}_{d_A,d_B}$ we can get values ε_A and ε_B to obtain similar privacy guarantees of two mechanisms $\mathcal{M}_A$ and $\mathcal{M}_B$. We note that previous work (Feyisetan et al., 2019) compared the privacy of two mechanisms looking at the probability that a mechanism returns the exact same word used as input. However, such notion does not always give a fair comparison between mechanisms, as that can vary considerably on embedding spaces. In contrast, our method is theoretically robust, with well defined bounds.

6. Experiments

In this section we compare BRR to previous work in terms of privacy, utility and efficiency. Since in DP we usually have a privacy-utility trade-off when applying mechanisms, we will fix the privacy of our mechanisms in order to compare their utility, using the methodology described in Section 5. Our experiments will use the IMDB dataset (Maas et al., 2011), with training data being 50% of the original training dataset, validation data being the other 50% of the original training data, and testing data being 50% of the original testing dataset. The experiments were performed on an AWS EC2 p3.2xlarge instance. More specific setup is described on the next paragraphs.

Privacy: In our experiments we vary ε for Madlib and set the ε for BRR using the privacy ratio. To have a more conservative approach, we use an average distance aggregation $\mathcal{P}_d^{avg}$ to fix privacy loss of BRR and Madlib. We note though that using $\mathcal{P}_d^{max}$ would give significantly better results for BRR in comparison to Madlib.

Utility: To evaluate the utility of the metric-DP mechanisms, we build ML models for sentiment analysis on training data privatized by each mechanism and compare the accuracy of the trained models on a separate testing dataset. For Madlib we use Euclidean distance on a fixed embedding space from GloVe (Pennington et al., 2014) with 300 dimensions. BRR starts from the same embedding, then transforms it into a binary representation with 256 dimensions as described in (Tissier et al., 2019). The sentiment classification models follow the FastText classifier (Joulin et al., 2017). The accuracy on the test dataset is shown in Figure 1a.



(a) Utility Evaluation

(b) MIA Evaluation

Figure 1. Comparison of mechanisms with 95% confidence interval over 10 independent trials for various ε. Baseline is built with models trained on original sensitive data. ε is first set for Madlib and transformed for BRR using the method from Section 5.

As seen in Figure 1a, in terms of utility, we obtain similar results for Madlib's ε under 5, and improved results for BRR for larger ε, where each setting is averaged over 5 independent trials, with the shaded are showing the 95% confidence interval.

Defense against attacks: We also include the results of a Membership Inference Attack (MIA) (Shokri et al., 2017), which tries to infer the presence of observations used to train a given model based only on black-box access, that is, the attacker is only able to query the deployed model and does not have access to specific weights. Lower score of the attack is better, representing more privacy preservation.

In summary, we have a *target* model that we can only query, e.g. through a public API, which we plan to attack to determine membership of particular users in its training set. To attack the target model we train a *shadow* model, based on

disjoint data we have available. This model tries to emulate the behavior of the target model. For example, we could try attacking a next-word prediction model trained on private data, by training a similar model using data from a user's public Twitter feed. Our dataset will then have two labels. The first called *member* we use to train the shadow model. The second, called *non-member* we set aside to train an *attack* model that will tell us if an example was part of the training data of the target model.

For this step, we use 50% of the original IMDB testing data to train a shadow model, another 50% to validate it. Models created for utility evaluation are the target for the attack. The attack model is an MLP with two layers having 64 hidden nodes each, with ReLU activations. Results are on Figure 1b. We can see that for the various privacy levels represented by ε tested, we obtain practical privacy protection, represented by the drop in AUC of the attack model. More specifically, we see that as we increase ε, which decreases the formal privacy guarantees of DP, we also obtain also less *empirical* privacy, represented by larger AUC. In this case, for very large $\varepsilon > 12$ we see a bigger impact on MIA for BRR. However, even though Madlib has smaller impact seen on MIA, that is not a *formal* privacy guarantee. Therefore, such large values of ε are not recommended for both mechanisms on the dataset analyzed. Finally, we note that for $\varepsilon \leq 12$, where both mechanisms have AUC of approximately 0.50 for MIA, we see BRR with similar or better utility than Madlib.

Efficiency: To evaluate how efficient the mechanisms are on different aspects, we consider the size of embedding vectors and index for approximate nearest neighbors, the wall time of privatization per word and total wall time per mechanism. For nearest neighbors search, we use FAISS (Johnson et al., 2017), a library that deals with both real and binary vectors.

For our relatively small vocabulary, the nearest neighbors index built for BRR achieved a compression rate of 97.9% (4MB vs. 200MB), while the vocabulary file, along with embeddings, was also 98.5% smaller (6MB vs. 300MB) compared to the ones used by Madlib.

For a fair time comparison, we looked for optimal ways of improving the computation of nearest neighbors during privatization of both BRR and Madlib. This step is the most time consuming of both methods and Madlib uses real-valued data with the Euclidean distance, while BRR uses binary data and the Hamming distance.

In our experiments, BRR was on average 68% faster to privatize a word compared to Madlib, as we can see on Figure 2. Using binary embedding vectors significantly improved the running time of our solution compared to using real-valued vectors, due to the possibility of using more efficient algorithms and hardware-level optimizations through binary

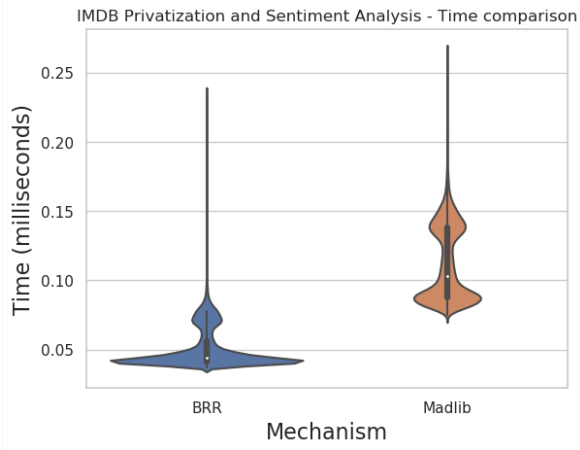


Figure 2. Wall-time comparison between BRR and Madlib on IMDB dataset.

operations. We used the FAISS library[2] that implements algorithms (Norouzi et al., 2012) tailored for binary data, for both exact and approximate nearest neighbors search. We note that FAISS has additional features in the library that we did not apply but could be further explored, such as product quantization, which is a technique for lossy compression of high-dimensional vectors, and PCA. Similarly, for Madlib we use approximate nearest neighbors, more specifically the Annoy library[3]. This speeds the retrieval at the cost of losing the guarantee to find the exact nearest neighbor.

7. Conclusion

We presented a mechanism for efficient text privatization on-device with formal differential privacy guarantees. We demonstrated that our new mechanism enables performing privatization on-device through the use of binary embeddings, providing zero-trust privacy for customers, while maintaining better or competitive utility than the state-of-the-art. As future work, we plan to implement improvements in the nearest neighbors search step, exploring smaller dimensionality of embedding vectors and including privatization of word context vectors for improved utility. We are currently exploring other DP mechanisms like using the exponential mechanism(McSherry and Talwar, 2007) that allows for the use of any metric function in the privatization process. Finally, a related line of work is how to achieve on-device privacy for tasks like Automatic Speech Recognition.

[2] https://github.com/facebookresearch/faiss

[3] https://github.com/spotify/annoy

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, Dec 2017. ISSN 2307-387X. doi: 10.1162/tacl_a_00051. URL http://dx.doi.org/10.1162/tacl_a_00051.

Konstantinos Chatzikokolakis, Miguel E Andrés, Nicolás Emilio Bordenabe, and Catuscia Palamidessi. Broadening the scope of differential privacy using metrics. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 82–102. Springer, 2013.

Thomas Diethe. Preserving privacy in analyses of textual data, 2020. URL https://www.amazon.science/blog/preserving-privacy-in-analyses-of-textual-data.

Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *Advances in Neural Information Processing Systems*, pages 3571–3580, 2017.

Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Theoretical Computer Science*, 9(3-4):211–407, 2014.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

Giulia Fanti, Vasyl Pihur, and Úlfar Erlingsson. Building a rappor with the unknown: Privacy-preserving learning of associations and data dictionaries. *Proceedings on Privacy Enhancing Technologies*, 2016(3):41–61, 2016.

Natasha Fernandes, Mark Dras, and Annabelle McIver. Author obfuscation using generalised differential privacy. *CoRR*, abs/1805.08866, 2018. URL http://arxiv.org/abs/1805.08866.

Oluwaseyi Feyisetan, Tom Diethe, and Thomas Drake. Leveraging hierarchical representations for preserving privacy and utility in text. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 210–219. IEEE, 2019.

Oluwaseyi Feyisetan, Borja Balle, Thomas Drake, and Tom Diethe. Privacy- and utility-preserving textual analysis via calibrated multivariate perturbations. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, WSDM '20, page 178–186, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450368223. doi: 10.1145/3336191.3371856. URL https://doi.org/10.1145/3336191.3371856.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, 2017. doi: 10.18653/v1/e17-2068. URL http://dx.doi.org/10.18653/V1/E17-2068.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 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, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-1015.

Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *FOCS*, volume 7, pages 94–103, 2007.

Mohammad Norouzi, Ali Punjani, and David J Fleet. Fast search in hamming space with multi-index hashing. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3108–3115. IEEE, 2012.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

Dinghan Shen, Pengyu Cheng, Dhanasekar Sundararaman, Xinyuan Zhang, Qian Yang, Meng Tang, Asli Celikyilmaz, and Lawrence Carin. Learning compressed sentence representations for on-device text processing. *arXiv preprint arXiv:1906.08340*, 2019.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

Apple Differential Privacy Team. Learning with privacy at scale. *Available at https://machinelearning.apple.com/2017/12/06/learning-with-privacy-at-scale.html*, 2017.

Julien Tissier, Christophe Gravier, and Amaury Habrard. Near-lossless binarization of word embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7104–7111, 2019.

Yue Wang, Xintao Wu, and Donghui Hu. Using randomized response for differential privacy preserving data collection. In *EDBT/ICDT Workshops*, volume 1558, pages 0090–6778, 2016.

Stanley L Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

A. Motivation for Privacy Ratio

In the standard definition of differential privacy (Dwork et al., 2006), the privacy guarantees provided by different mechanisms are compared by looking only at the ε value: mechanisms with same ε provide the same privacy guarantees. However, referring to Equation 1 we see that for metric-DP the privacy bound is directly impacted not only by ε but also by the distance metric. As a result, comparing mechanisms with different metrics is a non-trivial task. In this context we propose calculating, for each mechanism, a privacy loss bound for all $x, x' \in \mathcal{X}$.

To compare privacy among mechanisms first we look at the privacy loss, which is a general function defined for randomized mechanisms, not specific to any differential privacy definition.

Definition A.1. *Let $\mathcal{M} : \mathcal{X} \to \mathcal{Y}$ be a randomized mechanism with density function $p_{\mathcal{M}(x)}(y)$, then for any $x, x' \in \mathcal{X}$ and all outputs $y \in \mathcal{Y}$ the privacy loss function is defined as:*

$$\mathcal{L}_{\mathcal{M},x,x'}(y) = \ln\left(\frac{p_{\mathcal{M}(x)}(y)}{p_{\mathcal{M}(x')}(y)}\right) \tag{4}$$

Every differentially private mechanism will have by definition an upper bound for $\mathcal{L}_{\mathcal{M},x,x'}(y)$, and more specifically, metric-DP, as we show next.

Corollary A.1. *Given a randomized mechanism $\mathcal{M} : \mathcal{X} \to \mathcal{Y}$ that is εd-differentially private for a given distance function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$, we have that for any $x, x' \in \mathcal{X}$ and all outputs $y \in \mathcal{Y}$:*

$$\mathcal{L}_{\mathcal{M},x,x'}(y) < \varepsilon \cdot d(x, x') \tag{5}$$

From Equation 5 above we can see that the privacy loss bound of any pair x, x' depends on their distance and ε. Therefore, for a given metric d we propose calculating $d(x, x')$ for every possible pair $x, x' \in \mathcal{X}$ in order to obtain the overall bound on the privacy loss. With these distances, we can define a privacy measurement $\mathcal{P}_d$ for the maximum or average, as we now formalize.

Definition A.2. *For a given distance function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ and finite space $\mathcal{X}$, we define the privacy measurement $\mathcal{P}_d^{max}$ and $\mathcal{P}_d^{avg}$ as:*

$$\mathcal{P}_d^{max} = \max_{\forall x,x' \in \mathcal{X}} d(x, x') \tag{6}$$

$$\mathcal{P}_d^{avg} = \sum_{\forall x,x' \in \mathcal{X}} d(x, x')/|\mathcal{X}|^2 \tag{7}$$

For example, in the context of words as input, $\mathcal{P}_d$ can be calculated using a given distance metric on the embedding space for every word in the vocabulary. Comparing the two privacy measurements above, we note that $\varepsilon \cdot \mathcal{P}_d^{max}$ has the advantage of giving an overall bound on the privacy loss. This can be derived from Equations 5 and 6, such that for any $x, x' \in \mathcal{X}$ and all outputs $y \in \mathcal{Y}$, the privacy loss is bounded by:

$$\mathcal{L}_{\mathcal{M},x,x'}(y) < \varepsilon \cdot d(x, x') < \varepsilon \cdot \max_{\forall x,x' \in \mathcal{X}} d(x, x') = \varepsilon \cdot \mathcal{P}_d^{max} \tag{8}$$

On the other hand, $\mathcal{P}_d^{avg}$ gives average distance bounds over a given space $\mathcal{X}$, which is a more conservative approach and may be of interest to avoid spaces with outliers that individually would largely affect the privacy loss upper bound.

Given the above, $\varepsilon \cdot \mathcal{P}_d^{avg}$ or $\varepsilon \cdot \mathcal{P}_d^{max}$ can be used as estimates of privacy loss bounds, which allows us to fairly compare mechanisms using different distance metrics, as defined in Section 5 by equalizing their bounds through a privacy ratio.