Privacy of Textual Data: The pyPANTERA Package.*

Discussion Paper

Francesco Luigi, De Faveri[1], Guglielmo, Faggioli[1] and Nicola, Ferro[1]

[1]*Department of Information Engineering, University of Padova, Padova, Italy*

Abstract

Privacy is an essential aspect to consider when processing sensitive textual information in Natural Language Processing (NLP) and Information Retrieval (IR) tasks. Private medical records, queries, online posts and reviews can contain sensitive information that can endanger the confidentiality of users' data. To address this privacy issue, the gold-standard framework employed to protect such sensitive information when dealing with textual sentences is the ε-Differential Privacy (DP) obfuscation framework. However, to implement, develop and test state-of-the-art mechanisms, there is a need for a unified framework for such new obfuscation mechanisms. pyPANTERA is designed as a modular, extensible library developed to enrich DP techniques, enabling the integration of new DP mechanisms and allowing reproducible comparison of the current mechanisms. The effectiveness of the pyPANTERA package is measured by applying it to sentiment analysis and query obfuscation protocols. The library's source code is available in the public repository at https://github.com/Kekkodf/pypantera.

Keywords

Privacy Preserving Mechanisms, Differential Privacy, NLP, Information Retrieval, Security

1. Introduction

Natural Language Processing (NLP) and Information Retrieval (IR) systems are commonly developed and trained on textual data, e.g., queries, documents, and online posts, that contain sensitive and personal user information. Such a processing of textual data can pose privacy risks to the safety of users. For example, the queries a user submits to a search engine or the textual content that they can post on online social networks can contain personally identifiable information, e.g., the name or address of the searcher and details about the user's private sphere, e.g., political views, sexual orientation, that might expose them to blackmailing and cyber bullying [2] or even endanger their safety in illiberal countries [3, 4]. Consequently, the privacy research community [5, 6, 7, 8, 9] has stressed the importance of privacy for textual data analysis proposing different strategies of textual obfuscation. Such privatization techniques are based on the gold-standard definition of privacy, represented by ε-Differential Privacy (DP) [10]. The DP formal framework was introduced to provide users with the "Plausible Deniability" property, i.e., the outcome of any analysis is statistically indistinguishable considering a given privacy budget ε. A limitation within state-of-the-art obfuscation methodologies is that these approaches have been evaluated across different tasks and datasets; nevertheless, they have not been structured within a unified framework for text obfuscation in NLP and IR. Therefore, privacy practitioners can benefit from a compact, modular, and adaptable framework that encourages the rapid design of novel DP methodologies and permits a consistent and efficient evaluation against state-of-the-art techniques across multiple experimental tasks.

In this work, we discuss the pyPANTERA [1], an open-source unified, flexible and user-friendly framework for DP mechanisms implementation and comparison. Moreover, we bring together state-of-the-art mechanisms [11, 12, 13, 14, 15, 16] based on the DP framework and used for NLP and IR privacy tasks. pyPANTERA is structured into modules that implement different families of obfuscation

SEBD 2025: 33rd Symposium on Advanced Database Systems, June 16-19, 2025, Ischia, Italy

*This is an extended abstract of [1].

francescoluigi.defaveri@phd.unipd.it (F. L. De Faveri); faggioli@dei.unipd.it (G. Faggioli); ferro@dei.unipd.it (N. Ferro)

https://www.dei.unipd.it/~defaverifr/ (F. L. De Faveri); https://www.dei.unipd.it/~faggioli/ (G. Faggioli); https://www.dei.unipd.it/~ferro/ (N. Ferro)

0009-0005-8968-9485 (F. L. De Faveri); 0000-0002-5070-2049 (G. Faggioli); 0000-0001-9219-6239 (N. Ferro)

mechanisms, specifically sampling and embedding perturbation approaches, along with an evaluation module used to assess privacy and text the empirical correctness of the mechanisms. The obfuscation modules provide distinct interfaces for different mechanism families, ensuring the integration of new algorithms alongside existing ones. On the other hand, the evaluation module enables practitioners to assess the privacy of the obfuscated text by measuring the similarity between the original and obfuscated sentences and testing the effectiveness of the obfuscation mechanisms implemented.

Finally, we report the results of the implemented mechanisms, enforcing their use in real NLP and IR tasks and proving that those findings are comparable to those found in the original mechanism studies. This highlights the effectiveness of pyPANTERA as an important tool for privacy practitioners to implement prospective obfuscation techniques and accurately replicate results from prior studies. The code is open source under the GNU General Public License version 3.0 and publically available[1].

The paper is organized as follows: Section 2 describes other related works related to obfuscation techniques and tools publically available; moreover, Section 3 illustrates the design of the Python package, providing technical information about the resource, and finally Section 4 reports the results obtained from the tasks performed to evaluate the overall obfuscation framework.

2. Related Works

2.1. Background and Differential Privacy Approaches

Formal privacy is mathematically guaranteed by the definition of ε-Differential Privacy (DP) introduced by Dwork et al. [10]. A DP obfuscation mechanism $\mathcal{M}$ is an algorithm that receives as input a text and produces as output one or more noisy versions of the received input, regulating the amount of noise provided depending on the parameter $\varepsilon \in \mathbb{R}$, called *privacy budget* of the mechanism. An obfuscation mechanism $\mathcal{M}$ satisfy ε-DP if and only if, for any pair of neighbouring datasets D, D', i.e., datasets that differ for only one record, and given $\varepsilon > 0$, Equation 1 holds for all subsets $\mathcal{S} \subseteq \text{Image}(\mathcal{M})$.

$$\Pr\{\mathcal{M}(D) \in \mathcal{S}\} \leq e^{\varepsilon}\Pr\{\mathcal{M}(D') \in \mathcal{S}\} \tag{1}$$

Equation 1 grants the property of "plausible deniability" to the user: an adversary cannot confirm with absolute certainty the specific input (the user's original data) corresponding to a selected output.

However, to provide this property to textual data, the original definition of ε-DP is extended to metric spaces [17]. Once a text is encoded into a vector, Metric-DP [17] ensures that a randomized mechanism $\mathcal{M} : \mathbb{R}^n \to \mathbb{R}^n$ defined over a geometric space with distance function $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^+$ respects the definition of DP, iff, for any triplets of points $w, w', \hat{w} \in \mathbb{R}^n$, the inequality in Equation 2 is respected.

$$\Pr\{\mathcal{M}(w) = \hat{w}\} \leq e^{\varepsilon d(w,w')}\Pr\{\mathcal{M}(w') = \hat{w}\} \tag{2}$$

Obfuscation mechanism based on ε-DP and ε-Metric DP for natural texts has gained strong interests from the research and industry community [18, 5]. Specifically concerning these types of obfuscation mechanisms, the common categorization is based on the nature of the obfuscation perturbation applied to the texts. On the one hand, the mechanisms presented in [11, 12, 13] obfuscate the embeddings of the terms within the sentence by adding statistical noise following the privacy budget ε. Conversely, the mechanisms outlined in [14, 15, 16] rely on the initial computation of a score between word embeddings to rank analogous terms, utilizing ε to modify the probability of sampling the new words.

2.2. Differential Privacy Resources

Several endeavours are available to provide privacy to structured tabular data. Such libraries primarily facilitate the implementation of private statistical interrogation and private machine learning pipelines, such as the computation of Differentially Private Stochastic Gradient Descent [19, 20]. According to the evaluation proposed in [21, 22], examples of such libraries include IBM Diffprivlib [23], Meta

[1] https://github.com/Kekkodf/pypantera.

PyTorch Opacus [24], and Google TensorFlow Differential Privacy [25] toolkit. Furthermore, built as a forked project of Google TensorFlow Differential Privacy and OpenDP [26], OpenMined has released PyDP [27], a wrapper library in Python used for aggregating sensitive statistics across tabular datasets.

In NLP, text sanitization and anonymization are another privacy aspect. Text sanitization and anonymization concern removing sensitive data from textual data by substituting them with *placeholders* or censoring with random symbols. Microsoft Presidio [28] is constructed employing the SpaCy [29] library and consists of an Analyzer and an Anonymizer, which are designed to detect and mask personally identifiable information within a specified sentence. The Analyzer leverages regular expression rules and Named Entity Recognition Machine Learning models supplied by SpaCy to identify sensitive terms within the provided context. Thus, after the identification phase, Presidio employs the anonymization module to obfuscate such information by redacting, hashing, or replacing the identified sensitive data, generating an obfuscated censored version of the original text. Although Presidio and pyPANTERA operate on textual data, they address distinct privacy considerations: data sanitization and obfuscation, respectively. Therefore, these two resources can be considered complementary. In future work, we intend to merge the functionalities of Presidio and pyPANTERA to integrate Presidio's data identification and sanitization capabilities with the semantic obfuscation features of the DP framework in pyPANTERA within the obfuscation pipeline, thus designing an DP mechanism able to redact texts formally.

3. pyPANTERA

3.1. Obfuscation Pipeline

Figure 1 reports the general obfuscation pipeline of how the text is processed using pyPANTERA. The initial step involves the tokenization and parsing of the input text to eliminate punctuation while also converting all capitalized letters within the sentence to lowercase. The Initialization Phase concludes upon receiving the practitioner's selected parameters necessary to initialize the chosen obfuscation mechanism. The mechanisms available are either based on noisy embedding obfuscation strategies or on the noisy sampling of the terms employed in the obfuscated text produced. After each term in the sentence is finally obfuscated, all texts are reassembled in order to generate the user-required number of obfuscation variants. Such obfuscation produced is either stored in a single text or a suitable data frame and saved in a CSV file. Finally, the obfuscated versions of the texts can be used to perform the NLP and IR tasks privately. Additionally, pyPANTERA offers a module to assess the level of privacy granted.

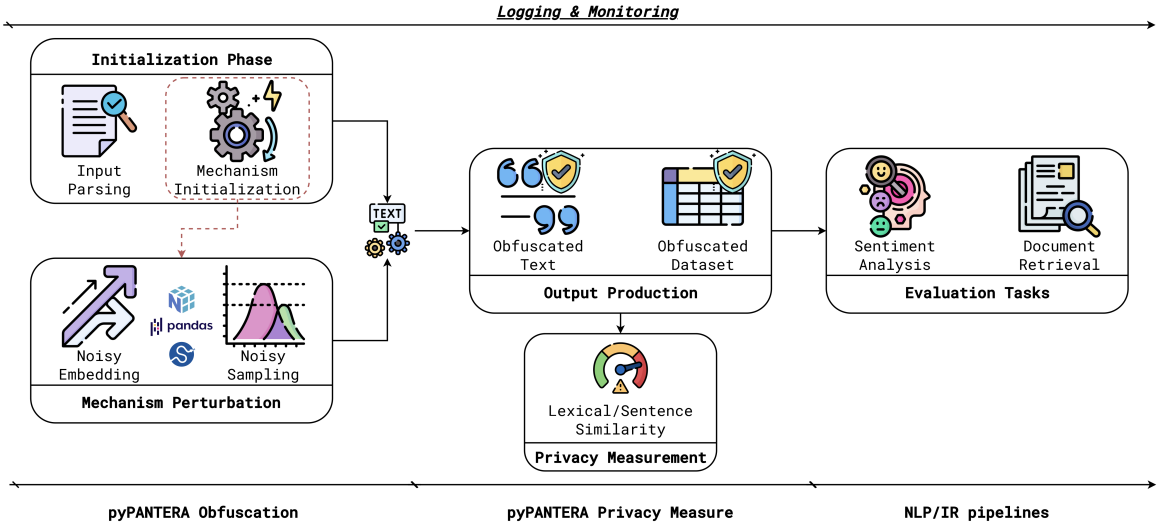

Figure 1: Pipeline of the pyPANTERA library.

3.2. Development Workflow

3.2.1. Requirements and Initial Usage

pyPANTERA is developed in Python (version 3.10) and requires Python ≥ 3.7 as the minimum version. Python was selected due to its accessibility, fast prototyping, and active user community. Moreover, as the tasks for which the obfuscation mechanisms are implemented depend on deep learning methods, ensuring rapid interoperability between obfuscation and the overall pipeline significantly enhances the efficiency of conducting experiments within the framework. The library can be installed and used in two ways: the first manner, i.e., the recommended one, is by cloning the repository of the resource available in GitHub[2]. The reason for the cloning is to ensure the last version of the mechanisms and methods. In addition, the README provides detailed instructions for setting up the virtual environment for conducting obfuscation and analysis. Alternatively, pyPANTERA can be installed using `pip` to download the package from PyPI[3], using the command `pip install pypantera`.

One of the advantages of pyPANTERA is that it is accessible to privacy practitioners of all expertise. To achieve this, pyPANTERA constructs upon popular data science libraries, i.e., Numpy [30], Pandas [31], and SciPy [32]. In addition, to optimize large amounts of text obfuscation, the library supports parallel computing with the Python library `multiprocessing`[4], increasing the efficiency.

3.2.2. Mechanisms Overview

New obfuscation mechanisms can be developed using the abstract classes provided by pyPANTERA. The library's UML diagram is accessible in the project repository, and it features a general abstract DP mechanism class for initializing new mechanisms. Additionally, distinct child abstract classes corresponding to embedding and sampling perturbation define each specific obfuscation process. An obfuscation mechanism has three main phases, i.e., Preprocessing, Distortion and Selection, depicted in Figure 2. The preprocessing phase deals with the tokenization and removal of alpha-numeric terms, after which an embedding model is usually employed to obtain the vectors of the terms in the original text. The second phase, i.e., the Distortion phase, modifies such term embeddings, considering the ε privacy budget and the other parameters specific to each mechanism. Finally, there is the Selection phase, where the final obfuscated word is selected to compose the produced privatized text.

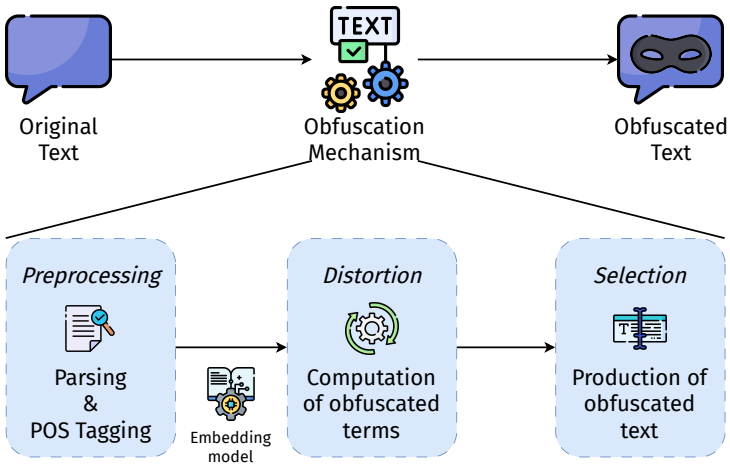

Figure 2: General steps of an obfuscation mechanism.

We report a list of state-of-the-art mechanisms available in the package. The mechanisms have been categorized into *Embedding* (Cumulative Multivariate Perturbation Mechanism (CMP), Mahalanobis

[2]https://github.com/Kekkodf/pypantera
[3]https://pypi.org/
[4]https://docs.python.org/3/library/multiprocessing.html

(Mhl), Vickrey CMP Mechanism (VickreyCMP), Vickrey Mahalanobis Mechanism (VickreyMhl)) and *Sampling* (Custumized Text Mechansim (CusText), Sanitization Text Mechanism (SanText), Truncated Exponential Mechanism (TEM)) perturbation groups to delineate the type of obfuscation process they perform. More details can be found in the repository.

- *Embedding Obfuscation*: Generally speaking, this family of obfuscation mechanisms is related to the alteration of the terms embeddings in the original text, adding a certain amount of statistical noise sampled proportionally to the ε privacy budget.
  - CMP [11]: After encoding each term in the original text, the statistical noise sampled from an n - dimensional Laplace distribution and it is added to the embedding of the terms. Finally, the new obfuscated terms are selected considering the proximity of the respective embeddings to the noisy ones computed.
  - Mhl [12]: Similarly to the CMP mechanisms, after the encoding of the terms in the input text, the noise is sampled from an n - dimensional Normal distribution proportional to the λ regularized Mahalanobis norm of the term embedding, stretching the obfuscation noise towards more similar terms, and the ε parameter. Finally, the selection of the new term is based on the proximity of the obfuscated embeddings to the original one.
  - VickreyCMP and VickreyMhl [13]: In this mechanism, the preprocessing and distortion with noise is defined by the parent method (CMP or Mhl) and the obfuscation term is then selected based on a free parameter threshold $t \in (0, 1)$.
- *Sampling Obfuscation*: While the Embedding obfuscation mainly considered the Distortion phase of an obfuscation mechanism, these strategies deal with the noisy selection of the obfuscated terms in the output texts. In this case, the mechanisms do not alter the embedding representation of the terms, thus missing the distortion phase in Figure 2.
  - CusText [15]: Selecting a new term involves a sampling approach, where the replacement word is chosen from a set of k possible term candidates. The determination of these candidates is based on their similarity to the original term, which is assessed through the distances between word embeddings. The k words with the highest similarity scores, i.e., lowest distances, are identified, from which one is selected exponentially proportional to ε.
  - SanText [14]: In this mechanism, there is no limitation of the top k most similar words, conversely with the CusText method, but all possible terms can be used.
  - TEM [16]: The noise, sampled from an n - dimensional Gumbel distribution, is incorporated into the score computed based on the distances between the vector embeddings. A truncation parameter β is introduced to limit the possible obfuscation candidates during the selection phase. Thus, the new term is chosen according to the maximum noisy score obtained by a term using the exponential mechanism [33] for sampling.

3.2.3. Functionalities

pyPANTERA enforces different utility functions to help the practitioner get an exhaustive view of all the pipeline steps. Therefore, the package offers an appropriate class to speed up the initialization of the embedding vocabulary that uses parallelization to read the embeddings from the supplied file. Moreover, using the `logging` python library[5] the method creates a folder containing a logger file to report all the action information regarding mechanism parameters, time of execution and steps executed. Finally, to evaluate the similarities between the original and obfuscated texts and thus assess the privacy provided to the texts, pyPANTERA implements the Jaccard Index to compute the overlapping terms, i.e., offering a proxy measure on the lexical similarity of the produced texts, and a cosine similarity among the contextual embeddings of the sentences depending on a configurable Transformer model, i.e., showcasing the sentence similarity between input and output texts.

[5] https://docs.python.org/3/library/logging.html

4. Experimental Evaluation

In this Section, we report the experiments performed to verify the effectiveness of the pyPANTERA package. As a downstream task, we employed the setups and methodology of the original studies, i.e., sentiment analysis, classification and document retrieval. Finally, to assess the levels of privacy provided, we followed the methodology proposed in [9] and computed the cosine similarity and Jaccard scores between original and obfuscated texts using the methods implemented in pyPANTERA.

4.1. Dataset and Experimental Setup

To evaluate the correctness and effectiveness of the pyPANTERA library, we conduct experiments using NLP tasks similar to those employed in the original state-of-the-art mechanism studies, specifically sentiment analysis. Additionally, following the methodology proposed by Faggioli and Ferro [9], we assess the library's robustness in implementing the query obfuscation pipeline for an IR task, i.e., document retrieval, while ensuring user privacy protection. For the sentiment analysis task, we used the Kaggle Twitter sentiment analysis[6] test set. On the other hand, in the document retrieval task, we obfuscated the queries from the TREC Deep Learning (DL'19) [34], based on the MSMARCO [35] passage corpus. Finally, we measured the privacy levels achieved by the former query collection using the metrics module in pyPANTERA. The default initialization parameters used to configure the obfuscation mechanisms are reported in Table 1. The privacy budget ε was selected to verify the impact of such parameter in a wide range of possible values, i.e., $\varepsilon \in \{1, 5, 10, 12.5, 15, 17.5, 20, 50\}$. To encode the texts, the default embeddings in the package and used for all the tasks are read from a local file containing the pre-computed vectors of GloVe [36] from Wikipedia 2014 publically available[7].

Table 1
Table of the parameters of the mechanism used to perform the different tasks in the experiments. Those parameters represents the default values with which the mechanisms are initialized.

Perturbation Families					
Embedding			*Sampling*		
Mechanism	**Constraints**	**Parameters**	**Mechanism**	**Constraints**	**Parameters**
CMP	-	-	CusText	$K \in \mathbb{N}$	$K = 10$
Mahalanobis	$\lambda \in [0, 1]$	$\lambda = 1$	SanText	-	-
VickreyCMP	$t \in [0, 1]$	$t = 0.75$	TEM	$\beta \in (0, 1)$	$\beta = 0.001$
VickreyMhl	$t \in [0, 1];\ \lambda \in [0, 1]$	$t = 0.75, \lambda = 1$			

A key feature of the pyPANTERA package is its flexibility in allowing practitioners to configure various parameters for the obfuscation mechanisms directly via command-line arguments. This functionality enables users to customize the behaviour of the mechanisms without modifying the underlying code. A practitioner can specify the desired mechanism and its parameters by executing a command such as:

```
python3 testObfuscationIR.py --mechanism VickreyCMP -t 0.5
```

In this example, the `–mechanism` flag selects the "VickreyCMP" obfuscation mechanism, while the `-t` parameter sets a threshold value of 0.5. This approach facilitates experimentation and fine-tuning, allowing users to efficiently adapt the obfuscation process to specific use cases.

4.2. Natural Language Processing

To demonstrate the capabilities of pyPANTERA, we conducted a standard NLP task—sentiment analysis—on a dataset of tweets collected from Twitter. For sentiment classification, we utilized the

[6] https://www.kaggle.com/datasets/jp797498e/twitter-entity-sentiment-analysis/data
[7] https://nlp.stanford.edu/projects/glove/

Twitter-roBERTa-base model, commonly referred to as TweetNLP [37], to extract sentiment from the preprocessed version of the tweets. As a performance metric, we measured accuracy in correctly identifying the sentiment labels of the tweets, aligning our evaluation with prior studies on obfuscation mechanisms [14, 16, 15]. This task aimed to demonstrate how the obfuscated tweets generated by pyPANTERA can be completely integrated into a basic NLP task, comparing different obfuscation techniques regarding their impact on model performance.

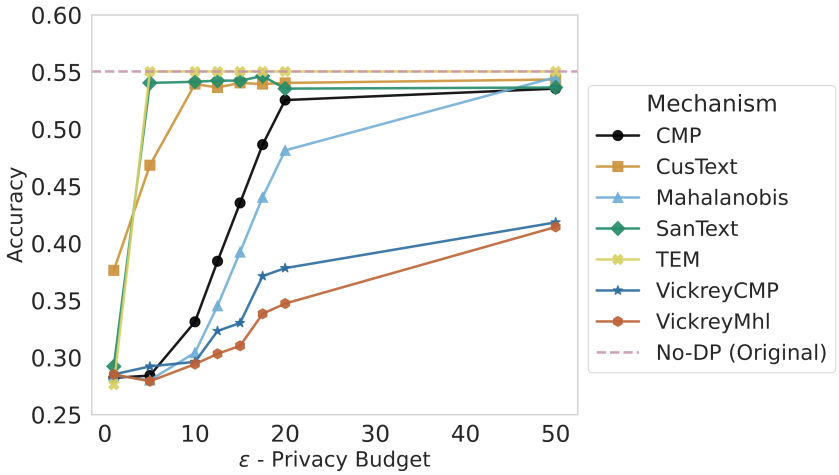

Figure 3: Mean Accuracy of the Sentiment Analysis task using the TweetNLP model [37], varying the privacy budget ε for the different mechanisms implemented in pyPANTERA.

Figure 3 presents the accuracy results as a function of the privacy budget ε for different obfuscation mechanisms. The findings are consistent with those reported in previous studies [14, 16, 15]. The TEM mechanism surpasses the CMP mechanism in sentiment classification, confirming the results obtained by Carvalho et al. [16]. Moreover, CusText performs better than SanText, aligning with the observations of Chen et al. [15]. In the context of noisy embeddings obfuscation, CMP and Mahalanobis exhibit a similar performance trend across different values of ε. In contrast, the Vickrey-based variants consistently demonstrate lower performance. The results highlight a clear distinction between the two obfuscation families: the sampling-based approach achieves higher precision for lower ε values, whereas the noisy embedding methods maintain lower performance under the same privacy constraints.

4.3. Information Retrieval and Privacy Analysis

Following the experimental methodology outlined by Faggioli and Ferro [9], we applied obfuscated MSMARCO DL'19 queries to retrieve relevant documents from the collection, ensuring user privacy during the retrieval process. Therefore, we re-ranked the retrieved results using the original (non-obfuscated) queries. For both retrieval and re-ranking, we utilized the Meta Contriever dense model [38]. The performance of the retrieval pipeline, measured in terms of Recall and nDCG@10, is presented in Table 2. Table 3 presents the similarity results, which quantify the relationship between the original and obfuscated DL'19 queries. Specifically, we evaluated two types of similarity using the metric functions available in pyPANTERA: lexical similarity, measured using the Jaccard index, and sentence-level similarity, computed as the cosine similarity between the contextual embeddings of the queries obtained from the Sentence-BERT MiniLM model [39]. In future versions of the library, we plan to implement new privacy measures like the one proposed in [40].

As observed by Faggioli and Ferro [9], and consistent with the theoretical expectations of a DP obfuscation mechanism, increasing the privacy budget ε results in enhanced performance of the obfuscation mechanism. However, this performance improvement comes with a weakening in the privacy guarantees, as illustrated in both Table 2 and Table 3. Moreover, the *Sampling* perturbation mechanisms tend to exhibit higher similarity between the original and obfuscated queries for lower values of ε compared to the *Embedding* perturbation mechanisms. This pattern suggests a trade-off

Table 2
Average Recall and nDCG@10 on the MSMARCO dl'19 collection [34] using the obfuscated queries for the searching process, and the original version for the reranking. The searching and the reranking process was performed using Contriver [38].

		Recall								**nDCG@10**							
		ε - Privacy Budget								ε - Privacy Budget							
Perturbation	**Mechanism**	1.0	5.0	10.0	12.5	15.0	17.5	20.0	50.0	1.0	5.0	10.0	12.5	15.0	17.5	20.0	50.0
Embedding	CMP	0.000	0.000	0.028	0.174	0.292	0.403	0.430	0.444	0.000	0.000	0.052	0.277	0.544	0.546	0.535	0.564
	Mahalanobis	0.000	0.000	0.001	0.077	0.134	0.290	0.368	0.447	0.000	0.000	0.003	0.103	0.262	0.455	0.494	0.565
	VickreyCMP	0.000	0.000	0.020	0.016	0.048	0.053	0.165	0.235	0.000	0.000	0.031	0.016	0.166	0.159	0.221	0.372
	VickreyMhl	0.000	0.001	0.002	0.002	0.029	0.042	0.122	0.191	0.000	0.005	0.007	0.004	0.062	0.097	0.158	0.293
Sampling	CusText	0.053	0.245	0.430	0.442	0.444	0.443	0.443	0.443	0.143	0.439	0.576	0.571	0.569	0.569	0.569	0.569
	SanText	0.000	0.444	0.447	0.448	0.444	0.450	0.447	0.444	0.000	0.564	0.569	0.570	0.568	0.559	0.568	0.562
	TEM	0.000	0.498	0.498	0.498	0.498	0.498	0.498	0.498	0.000	0.636	0.636	0.636	0.636	0.636	0.636	0.636
None	Original	-	-	-	-	-	-	-	-	0.498	-	-	-	-	-	-	-	-	0.636

Table 3
Average Lexical and Sentence similarity between the original and obfuscated queries of the MSMARCO dl'19 collection [34]. Lexical and Semantic similarity are computed using the implemented `metrics` module in pyPANTERA.

		Lexical Similarity *(Jaccard Similarity)*								**Semantic Similarity** *(MiniLM [39])*							
		ε - Privacy Budget								ε - Privacy Budget							
Perturbation	**Mechanism**	1.0	5.0	10.0	12.5	15.0	17.5	20.0	50.0	1.0	5.0	10.0	12.5	15.0	17.5	20.0	50.0
Embedding	CMP	0.000	0.000	0.119	0.274	0.460	0.735	0.785	0.935	0.025	0.037	0.225	0.429	0.628	0.836	0.847	0.902
	Mahalanobis	0.000	0.002	0.047	0.140	0.302	0.457	0.590	0.935	0.016	0.027	0.088	0.242	0.435	0.587	0.730	0.908
	VickreyCMP	0.000	0.000	0.039	0.061	0.180	0.191	0.164	0.212	0.018	0.045	0.103	0.169	0.348	0.382	0.435	0.596
	VickreyMhl	0.000	0.013	0.028	0.038	0.098	0.134	0.117	0.151	0.037	0.030	0.078	0.109	0.202	0.264	0.303	0.498
Sampling	CusText	0.089	0.374	0.816	0.880	0.925	0.929	0.929	0.935	0.357	0.627	0.881	0.900	0.908	0.908	0.909	0.910
	SanText	0.000	0.935	0.935	0.935	0.935	0.935	0.935	0.935	0.031	0.902	0.906	0.910	0.917	0.900	0.902	0.907
	TEM	0.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.037	1.000	1.000	1.000	1.000	1.000	1.000	1.000

between privacy and the precision of obfuscation, warranting further analysis and investigation in future work. We leave this observation as an open issue for future experiments.

5. Conclusions

Given the increasing concerns surrounding data confidentiality in textual analysis, privacy remains an important research domain for NLP and IR. In this paper, we introduced the functionality of pyPANTERA introduced in [1], a highly adaptable and extensible framework designed to systematically evaluate and compare different DP obfuscation mechanisms. Our framework significantly contributes to the privacy-preserving research community by establishing a well-defined and user-friendly text obfuscation pipeline, facilitating the development and integration of novel obfuscation techniques by researchers and practitioners in the privacy research field. The pyPANTERA library encompasses diverse functionalities, including real-time monitoring of the obfuscation process and a list of evaluation metrics to assess the level of privacy preserved beyond the formal analysis of the ε privacy budget. Furthermore, we conduct an extensive empirical analysis across standard NLP and IR tasks, demonstrating the effectiveness of pyPANTERA in providing a robust and unified environment for the comparative assessments of different obfuscation strategies based on DP. As part of future research, we aim to broaden the number of available obfuscation mechanisms in the framework and enhance the privacy evaluation module by introducing additional metric functions to refine the assessment of privacy guarantees.

Declaration on Generative AI

During the preparation of this work, the authors used Grammarly for Readability and Spelling checks. After using this tool, they reviewed and edited the content as needed and took full responsibility for the publication's content.

References

[1] F. L. De Faveri, G. Faggioli, N. Ferro, pyPANTERA: A python PAckage for Natural language obfuscaTion Enforcing pRivacy & Anonymization, in: E. Serra, F. Spezzano (Eds.), Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, CIKM 2024, Boise, ID, USA, October 21-25, 2024, ACM, 2024, pp. 5348–5353. URL: https://doi.org/10.1145/3627673.3679173. doi:10.1145/3627673.3679173.

[2] N. Chetty, S. Alathur, Hate speech review in the context of online social networks, Aggression and Violent Behavior 40 (2018) 108–118. URL: https://www.sciencedirect.com/science/article/pii/S1359178917301064. doi:https://doi.org/10.1016/j.avb.2018.05.003.

[3] H. Le, R. Maragh, B. Ekdale, A. High, T. Havens, Z. Shafiq, Measuring political personalization of google news search, in: The World Wide Web Conference, WWW '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 2957–2963. URL: https://doi.org/10.1145/3308558.3313682. doi:10.1145/3308558.3313682.

[4] E. Mustafaraj, E. Lurie, C. Devine, The case for voter-centered audits of search engines during political elections, in: Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency, FAT* '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 559–569. URL: https://doi.org/10.1145/3351095.3372835. doi:10.1145/3351095.3372835.

[5] Y. Zhao, J. Chen, A survey on differential privacy for unstructured data content, ACM Comput. Surv. 54 (2022). URL: https://doi.org/10.1145/3490237. doi:10.1145/3490237.

[6] I. Habernal, When differential privacy meets NLP: the devil is in the detail, in: M. Moens, X. Huang, L. Specia, S. W. Yih (Eds.), Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021, Association for Computational Linguistics, 2021, pp. 1522–1528. URL: https://doi.org/10.18653/v1/2021.emnlp-main.114. doi:10.18653/V1/2021.EMNLP-MAIN.114.

[7] I. Habernal, F. Mireshghallah, P. Thaine, S. Ghanavati, O. Feyisetan, Privacy-preserving natural language processing, in: F. M. Zanzotto, S. Pradhan (Eds.), Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: Tutorial Abstracts, Association for Computational Linguistics, Dubrovnik, Croatia, 2023, pp. 27–30. URL: https://aclanthology.org/2023.eacl-tutorials.6. doi:10.18653/v1/2023.eacl-tutorials.6.

[8] N. Carlini, F. Tramèr, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, Ú. Erlingsson, A. Oprea, C. Raffel, Extracting training data from large language models, in: 30th USENIX Security Symposium (USENIX Security 21), USENIX Association, 2021, pp. 2633–2650. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/carlini-extracting.

[9] G. Faggioli, N. Ferro, Query obfuscation for information retrieval through differential privacy, in: N. Goharian, N. Tonellotto, Y. He, A. Lipani, G. McDonald, C. Macdonald, I. Ounis (Eds.), Advances in Information Retrieval, Springer Nature Switzerland, Cham, 2024, pp. 278–294.

[10] C. Dwork, F. McSherry, K. Nissim, A. Smith, Calibrating noise to sensitivity in private data analysis, in: S. Halevi, T. Rabin (Eds.), Theory of Cryptography, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 265–284.

[11] O. Feyisetan, B. Balle, T. Drake, T. 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, Association for Computing Machinery, New York, NY, USA, 2020, pp. 178–186. URL: https://doi.org/10.1145/3336191.3371856. doi:10.1145/3336191.3371856.

[12] Z. Xu, A. Aggarwal, O. Feyisetan, N. Teissier, A differentially private text perturbation method using regularized mahalanobis metric, in: O. Feyisetan, S. Ghanavati, S. Malmasi, P. Thaine (Eds.), Proceedings of the Second Workshop on Privacy in NLP, Association for Computational Linguistics, Online, 2020, pp. 7–17. URL: https://aclanthology.org/2020.privatenlp-1.2.pdf. doi:10.18653/v1/2020.privatenlp-1.2.

[13] Z. Xu, A. Aggarwal, O. Feyisetan, N. Teissier, On a utilitarian approach to privacy preserving text generation, in: O. Feyisetan, S. Ghanavati, S. Malmasi, P. Thaine (Eds.), Proceedings of the Third

Workshop on Privacy in Natural Language Processing, Association for Computational Linguistics, Online, 2021, pp. 11–20. URL: https://aclanthology.org/2021.privatenlp-1.2. doi:10.18653/v1/2021.privatenlp-1.2.

[14] X. Yue, M. Du, T. Wang, Y. Li, H. Sun, S. S. M. Chow, Differential privacy for text analytics via natural text sanitization, in: C. Zong, F. Xia, W. Li, R. Navigli (Eds.), Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, Association for Computational Linguistics, Online, 2021, pp. 3853–3866. URL: https://aclanthology.org/2021.findings-acl.337. doi:10.18653/v1/2021.findings-acl.337.

[15] S. Chen, F. Mo, Y. Wang, C. Chen, J.-Y. Nie, C. Wang, J. Cui, A customized text sanitization mechanism with differential privacy, in: A. Rogers, J. Boyd-Graber, N. Okazaki (Eds.), Findings of the Association for Computational Linguistics: ACL 2023, Association for Computational Linguistics, Toronto, Canada, 2023, pp. 5747–5758. URL: https://aclanthology.org/2023.findings-acl.355. doi:10.18653/v1/2023.findings-acl.355.

[16] R. S. Carvalho, T. Vasiloudis, O. Feyisetan, K. Wang, TEM: High Utility Metric Differential Privacy on Text, "2023", pp. 883–890. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611977653.ch99. doi:10.1137/1.9781611977653.ch99. arXiv:https://epubs.siam.org/doi/pdf/10.1137/1.9781611977653.ch99.

[17] K. Chatzikokolakis, M. E. Andrés, N. E. Bordenabe, C. Palamidessi, Broadening the scope of differential privacy using metrics, in: E. De Cristofaro, M. Wright (Eds.), Privacy Enhancing Technologies, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 82–102.

[18] O. Klymenko, S. Meisenbacher, F. Matthes, Differential privacy in natural language processing the story so far, in: O. Feyisetan, S. Ghanavati, P. Thaine, I. Habernal, F. Mireshghallah (Eds.), Proceedings of the Fourth Workshop on Privacy in Natural Language Processing, Association for Computational Linguistics, Seattle, United States, 2022, pp. 1–11. URL: https://aclanthology.org/2022.privatenlp-1.1. doi:10.18653/v1/2022.privatenlp-1.1.

[19] M. Abadi, A. Chu, I. J. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, L. Zhang, Deep learning with differential privacy, in: E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, S. Halevi (Eds.), Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016, ACM, 2016, pp. 308–318. URL: https://doi.org/10.1145/2976749.2978318. doi:10.1145/2976749.2978318.

[20] N. Carlini, C. Liu, U. Erlingsson, J. Kos, D. Song, The secret sharer: evaluating and testing unintended memorization in neural networks, in: Proceedings of the 28th USENIX Conference on Security Symposium, SEC'19, USENIX Association, USA, 2019, p. 267–284.

[21] S. Zhang, A. Hagermalm, S. Slavnic, E. M. Schiller, M. Almgren, Evaluation of open-source tools for differential privacy, Sensors 23 (2023) 6509. URL: https://doi.org/10.3390/s23146509. doi:10.3390/S23146509.

[22] I. C. Ngong, B. Stenger, J. P. Near, Y. Feng, Evaluating the usability of differential privacy tools with data practitioners, 2024. arXiv:2309.13506.

[23] N. Holohan, S. Braghin, P. Mac Aonghusa, K. Levacher, Diffprivlib: the IBM differential privacy library, ArXiv e-prints 1907.02444 [cs.CR] (2019).

[24] A. Yousefpour, I. Shilov, A. Sablayrolles, D. Testuggine, K. Prasad, M. Malek, J. Nguyen, S. Ghosh, A. Bharadwaj, J. Zhao, G. Cormode, I. Mironov, Opacus: User-friendly differential privacy library in pytorch, CoRR abs/2109.12298 (2021). URL: https://arxiv.org/abs/2109.12298. arXiv:2109.12298.

[25] P. Subramani, N. Vadivelu, G. Kamath, Enabling fast differentially private SGD via just-in-time compilation and vectorization, in: M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, J. W. Vaughan (Eds.), Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual, 2021, pp. 26409–26421. URL: https://proceedings.neurips.cc/paper/2021/hash/ddf9029977a61241841edeae15e9b53f-Abstract.html.

[26] M. Gaboardi, M. Hay, S. Vadhan, A programming framework for opendp, Manuscript, May (2020).

[27] OpenMinded, Pydp, 2021. URL: https://github.com/OpenMined/PyDP, accessed: 2024.

[28] O. Mendels, C. Peled, N. Vaisman Levy, T. Rosenthal, L. Lahiani, et al., Microsoft Presidio: Context

aware, pluggable and customizable pii anonymization service for text and images, 2018.
[29] M. Honnibal, I. Montani, S. Van Landeghem, A. Boyd, spaCy: Industrial-strength Natural Language Processing in Python (2020). doi:10.5281/zenodo.1212303.
[30] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, T. E. Oliphant, Array programming with NumPy, Nature 585 (2020) 357–362. URL: https://doi.org/10.1038/s41586-020-2649-2. doi:10.1038/s41586-020-2649-2.
[31] Wes McKinney, Data Structures for Statistical Computing in Python, in: Stéfan van der Walt, Jarrod Millman (Eds.), Proceedings of the 9th Python in Science Conference, 2010, pp. 56 – 61. doi:10.25080/Majora-92bf1922-00a.
[32] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, SciPy 1.0 Contributors, SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python, Nature Methods 17 (2020) 261–272. doi:10.1038/s41592-019-0686-2.
[33] F. McSherry, K. Talwar, Mechanism design via differential privacy, in: 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings, IEEE Computer Society, 2007, pp. 94–103. URL: https://doi.org/10.1109/FOCS.2007.41. doi:10.1109/FOCS.2007.41.
[34] N. Craswell, B. Mitra, E. Yilmaz, D. Campos, E. M. Voorhees, Overview of the TREC 2019 deep learning track, CoRR abs/2003.07820 (2020). URL: https://arxiv.org/abs/2003.07820. arXiv:2003.07820.
[35] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, L. Deng, MS MARCO: A human generated machine reading comprehension dataset, in: T. R. Besold, A. Bordes, A. S. d'Avila Garcez, G. Wayne (Eds.), Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016, volume 1773 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2016.
[36] J. Pennington, R. Socher, C. Manning, GloVe: Global vectors for word representation, in: A. Moschitti, B. Pang, W. Daelemans (Eds.), Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Doha, Qatar, 2014, pp. 1532–1543. URL: https://aclanthology.org/D14-1162. doi:10.3115/v1/D14-1162.
[37] J. Camacho-collados, K. Rezaee, T. Riahi, A. Ushio, D. Loureiro, D. Antypas, J. Boisson, L. Espinosa Anke, F. Liu, E. Martínez Cámara, TweetNLP: Cutting-edge natural language processing for social media, in: W. Che, E. Shutova (Eds.), Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Association for Computational Linguistics, Abu Dhabi, UAE, 2022, pp. 38–49. URL: https://aclanthology.org/2022.emnlp-demos.5. doi:10.18653/v1/2022.emnlp-demos.5.
[38] G. Izacard, M. Caron, L. Hosseini, S. Riedel, P. Bojanowski, A. Joulin, E. Grave, Unsupervised dense information retrieval with contrastive learning, Trans. Mach. Learn. Res. 2022 (2022). URL: https://openreview.net/forum?id=jKN1pXi7b0.
[39] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2019. URL: https://arxiv.org/abs/1908.10084.
[40] F. L. De Faveri, G. Faggioli, N. Ferro, Measuring actual privacy of obfuscated queries in information retrieval, in: Advances in Information Retrieval: 47th European Conference on Information Retrieval, ECIR 2025, Lucca, Italy, April 6–10, 2025, Proceedings, Part I, Springer-Verlag, Berlin, Heidelberg, 2025, p. 49–66. URL: https://doi.org/10.1007/978-3-031-88708-6_4. doi:10.1007/978-3-031-88708-6_4.