Delft University of Technology

RedactBuster

Entity Type Recognition from Redacted Documents

Beltrame, Mirco; Conti, Mauro; Guglielmin, Pierpaolo; Marchiori, Francesco; Orazi, Gabriele

DOI
10.1007/978-3-031-70890-9_23

Publication date
2024

Document Version
Final published version

Published in
Computer Security – ESORICS 2024 - 29th European Symposium on Research in Computer Security, Proceedings

Citation (APA)
Beltrame, M., Conti, M., Guglielmin, P., Marchiori, F., & Orazi, G. (2024). RedactBuster: Entity Type Recognition from Redacted Documents. In J. Garcia-Alfaro, R. Kozik, M. Choraś, & S. Katsikas (Eds.), *Computer Security – ESORICS 2024 - 29th European Symposium on Research in Computer Security, Proceedings* (pp. 451-470). (Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Vol. 14983 LNCS). Springer. https://doi.org/10.1007/978-3-031-70890-9_23

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

Green Open Access added to TU Delft Institutional Repository

'You share, we take care!' - Taverne project

https://www.openaccess.nl/en/you-share-we-take-care

Otherwise as indicated in the copyright section: the publisher is the copyright holder of this work and the author uses the Dutch legislation to make this work public.

RedactBuster: Entity Type Recognition from Redacted Documents

Mirco Beltrame[1], Mauro Conti[1,2], Pierpaolo Guglielmin[1], Francesco Marchiori[1(✉)], and Gabriele Orazi[1,3]

[1] Department of Mathematics, University of Padua, Padua, Italy
{mirco.beltrame.1,pierpaolo.guglielmin}@studenti.unipd.it, mauro.conti@unipd.it, {francesco.marchiori.4,gabriele.orazi}@phd.unipd.it
[2] Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Delft, The Netherlands
[3] FDM Business Services, Milan, Italy

Abstract. The widespread exchange of digital documents in various domains has resulted in abundant private information being shared. This proliferation necessitates redaction techniques to protect sensitive content and user privacy. While numerous redaction methods exist, their effectiveness varies, with some proving more robust than others. As such, the literature proposes several deanonymization techniques, raising awareness of potential privacy threats. However, while none of these methods are successful against the most effective redaction techniques, these attacks only focus on the anonymized tokens and ignore the sentence context.

In this paper, we propose **RedactBuster**, the first deanonymization model using sentence context to perform Named Entity Recognition on redacted text. Our methodology leverages fine-tuned state-of-the-art Transformers and Deep Learning models to determine the anonymized entity types in a document. We test RedactBuster against the most effective redaction technique and evaluate it using the publicly available Text Anonymization Benchmark (TAB). Our results show accuracy values up to 0.985 regardless of the document nature or entity type. In raising awareness of this privacy issue, we propose a countermeasure we call *character evasion* that helps strengthen the secrecy of sensitive information. Furthermore, we make our model and testbed open-source to aid researchers and practitioners in evaluating the resilience of novel redaction techniques and enhancing document privacy.

Keywords: Document Redaction · Privacy · Personally Identifiable Information · Named Entity Recognition · Information Leakage

1 Introduction

Increasing document digitalization efforts have been adopted in several domains, such as the corporate sector, healthcare, and government [39]. These procedures allow for streamlined workflows, improving processing times and reducing

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024
J. Garcia-Alfaro et al. (Eds.): ESORICS 2024, LNCS 14983, pp. 451–470, 2024.
https://doi.org/10.1007/978-3-031-70890-9_23

reliance on manual manipulation. As a side effect, the amount of data exchanged has also seen a significant rise [29]. To illustrate the magnitude of the phenomenon, in 2022 and in Italy alone, the car and motorcycle insurance market generated more than 39 million documents between contracts and claims for a market of about 22.6 billion euro [19]. Each of these 39 billion documents contains sensitive information that needs to be protected. While the process of digitalization can improve efficiency and optimization in specific tasks thanks to the recent advancements of Artificial Intelligence (AI) and data-driven approaches, it can cause privacy concerns. Indeed, the lack of obfuscation of sensitive content may lead to unfair profiling of certain individuals or discrimination of specific groups [17]. For these reasons, *anonymization* defined as "the process of rendering personal data anonymous" has been mandated for General Data Protection Regulation (GDPR) compliance in the EU [14].

Several anonymization techniques have been proposed, given the importance of protecting user privacy and complying with regulations. In the context of privacy protection, *redaction* refers to the process of selectively editing or obscuring sensitive information from documents to prevent unauthorized access or disclosure [2]. It allows for removing only specific parts of a sentence while preserving the overall content of the text. In the Italian case, document redaction is required for all public administrations. The Personal Data Protection Guarantor (GPDP) obliges public organizations to publish how public resources were used by anonymizing Personally Identifiable Information (PII) that refers to individuals. However, this anonymization process might leave some information about the users if only sensitive words are redacted. For example, inferring the entity type of redacted content can pose a privacy threat as it could be cross-referenced with other open-source intelligence. This can also lead to security risks, as the knowledge of specific locations or individuals involved in sensitive documents could be leveraged for malicious purposes. Some examples of redaction techniques are shown in Fig. 1. One redaction technique blurs a specific text part to make it unreadable (Fig. 1b). Another considers specific parts of the document as images and pixelates them by reducing their image quality (Fig. 1c). Blackout or whiteout redaction techniques (when applied correctly) are more effective, as they allow to discard entirely the content to be anonymized and substitute it with black or white boxes (Fig. 1d).

While redaction techniques are commonly used in many domains, their security in protecting user privacy is uncertain. Indeed, several works in the literature propose attacks for determining redacted content in sensitive documents. For example, blurred and pixelated content can be unmasked with high accuracy [18]. For these reasons, other redaction techniques, such as blackout and whiteout, should be preferable. While the effectiveness of other redaction techniques has not yet been discussed, no attacks in the literature have been proposed. Furthermore, the existing attacks focus exclusively on the redacted tokens. Since with more effective anonymization techniques the sensitive content is erased from the document, this strategy is not effective anymore.

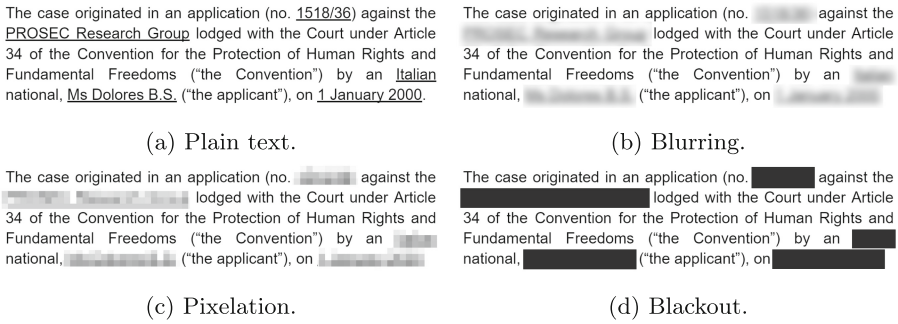

Fig. 1. Examples of redaction techniques.

Contribution. In this paper, we present **RedactBuster**, the first document deanonymization attack against the most effective redaction technique. We open-source and evaluate our framework on the most comprehensive dataset, obtaining results of up to 0.985. We also discuss the implementation of several countermeasures to our attack and propose a novel technique to enhance redaction efforts. Our contributions can be summarized as follows.

- We present **RedactBuster**, the first deanonymization attack against the most effective redaction technique. Our methodology leverages state-of-the-art Machine Learning (ML) and Deep Learning (DL) models for computing sentence embeddings and performing classification.
- We evaluate our framework on Text Anonymization Benchmark (TAB), the most extensive dataset publicly available on document redaction [32]. This dataset contains 1268 court cases in English, which have been properly annotated, labeled, and redacted. Our results show baseline accuracy values of 0.958, which can reach 0.985 by fine-tuning the embedding model on our specific task. We test several ML and DL models, comprehensively evaluating different architectures' capabilities and limitations.
- We propose *character evasion* as an effective countermeasure against our attack and test them against our model. This technique involves the exchange of specific homoglyphs, allowing users to read and process the document effortlessly but preventing malicious parties from extracting the redacted entity types. We also demonstrate the effectiveness of our countermeasure on the dataset and show how swapping only five characters can decrease the attack success rate to 0.195.
- We open-source our framework at: https://github.com/spritz-group/RedactBuster.

Organization. Our paper is organized as follows. In Sect. 2, we discuss related works on anonymization and attacks on redaction techniques. Section 3 presents our system and threat model, while technical details on the methodology are provided in Sect. 4. In Sect. 5, we evaluate our framework, and we propose countermeasures in Sect. 6. Finally, Sect. 7 concludes our work.

2 Related Works

The importance of redaction for protecting documents containing sensitive data or PII does not have a widely shared standard among parties to date. In June 2023, a litigation between Microsoft and the Federal Trade Commission demonstrated that *(i)* documents can still be redacted by hand using a Sharpie marker and *(ii)* redacted document represents a threat to organizations because of possible and expensive data leaks[1]. This event highlights how sensitive information circulating through documents is still subject to human error today. The unstructured nature of a written document makes it impossible to apply analytical methods to arrive at a specific result. Despite the current tools to support this problem, manual validation is simultaneously the only solution to curb the deficiency and the source of possible issues. Therefore, in this section, we describe which are the current solutions for document redaction, how PII can be recognized within a text, and which unredaction techniques have been found to date.

Redaction Techniques. Many studies and patents tackle the redaction phase per se, covering some parts of the text. In [21], the procedure is specifically crafted to work in synergy with Microsoft Office Word. In [27] a rule-set based automatic redaction system is presented, while [12] proposes a dynamic redaction based on tags specified by the user. Instead of covering, random generation of sensitive characters and numbers has also been patented [26]. A similar approach is presented in [1], which includes a redaction and storage system for redacted documents. Another valuable approach is presented in [33], which implies the concept of blockchain to preserve the security of an already-redacted document. Unfortunately, such studies neglect the core of the redaction process, which is identifying the sensitive characters of a text that must be covered.

Named Entity Recognition (NER). NER is a crucial component of Natural Language Processing (NLP). It identifies predefined categories of objects in text, such as names of individuals, organizations, locations, expressions of time, and quantities. In [10], the authors present a novel approach in the field by employing bidirectional Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN) architecture. The insight of the latter work influenced the creation of the well-known BERT [13], a milestone on which much work in NLP is still based today. BERT has also proven to be highly effective in developing techniques for NER [24,25]. Nevertheless, BERT models are also adopted as hyper-specialized models for particular domains such as [40] materials and aerospace engineering [36]. The scientific community has shown approaches for automation of NER and subsequent redaction of sensitive parts, albeit supervised by an end user [2]. On the same line, Microsoft developed and maintains an open source tool called PRESIDIO [28], which analyzes a text and, based on multiple recognizers, can detect and anonymize PII.

[1] https://www.theverge.com/2023/6/28/23777298/sony-ftc-microsoft-confidential-documents-marker-pen-scanner-oops.

Unredaction. Despite exciting developments in text redaction, studies are pointing out that some methods of information coverage are ineffective and vulnerable to possible data leaks. In [5], the authors assess standard PDF redaction tools vulnerability. The study enlightens the possibility of recovering first and last names just because of subpixel-sized shifts of characters. Instead, the authors of [18] demonstrate the inadequacy of mosaicing, blurring, and pixelating methods as graphical coverage of sensitive characters. To further demonstrate how dangerous such methods are, the security engineer Dan Petro released a proof-of-concept [31]. Although the outcome might be preferred to solid black rectangles, such aberrations represent an actual information leakage for a potential attacker.

3 System and Threat Model

We now discuss the assumptions defining both the redaction system's functionality and potential attackers' capabilities. In particular, Sect. 3.1 delves into the system model and the functionalities of a redaction pipeline in an adversary-free environment. In Sect. 3.2, we analyze the attacker's knowledge of the target model in real-world scenarios.

3.1 System Model

Digital documents are commonly used online to share knowledge, sign contracts, or write reports. Due to the nature of documents, it is expected to share such files while preserving the sensitive information. *Redaction* is the process that allows companies and organizations to identify and protect the most sensitive parts of a document by covering such portions of the text. To date, no objective standard has been certified as the best methodology for digital document redaction. The security of such an operation falls in the hands of individuals or companies that may adopt more or less virtuous methods.

A general pipeline that organizations adopt is the one shown in Fig. 2 and includes the following main steps:

1. *Automated Entity Recognition.* The text in a document is fed into a NER software, which produces an intermediate document version with highlighted entities that will potentially be redacted. Entities can be differentiated by using different highlighting colors depending on the kind of entity (e.g., proper name, date, location). If needed, entities that can or must be disclosed safely can be included in a whitelist.
2. *Human validation.* An operator takes charge of the intermediate document and manually reviews the highlighted entities to assess that there are no missed detections. Once the validation is complete, the final redacted version of the document can be generated.

Although redaction is a lossy process of the original digital document, many details can still be used to empower inference techniques for disclosure purposes.

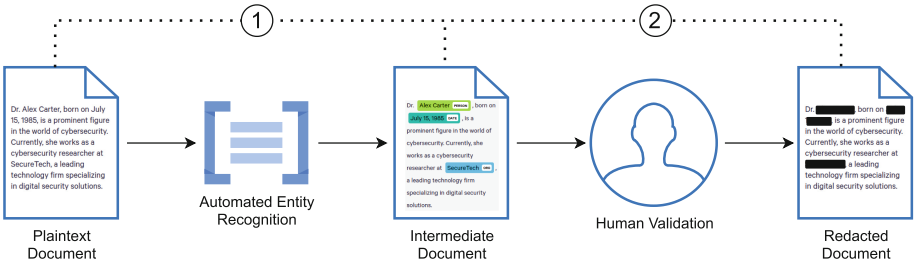

Fig. 2. General pipeline for document redaction.

3.2 Threat Model

RedactBuster aims to predict which kinds of entities are hidden underneath a redacted text. Even though working with unstructured text contained in documents is usually a challenge, our model wants to leverage the complexity of sentences to achieve its goal.

The attacker employing RedactBuster is interested in reconstructing the redacted text, ideally being able to rebuild the original unredacted document. In other words, they want to revert the pipeline described in Sect. 3.1. Documents are redacted to be shared with a broader audience that might not otherwise view the information in plain text. For this reason, we assume that the attacker has access to a redacted document that, if deanonymized, could lead to real gain. For instance, the attacker may want to access plaintext documents to extort, threaten, or ransom the victim.

Formal Definition. We will now outline the context in which the attacker operates, considering four key criteria [3]: their *goals*, *knowledge*, *capabilities*, and *strategies*.

- *Goal:* The attacker wants to recover the original version of a document that has been redacted.
- *Knowledge:* The attacker knows the area in which the document of interest falls. A list of examples includes fiscal, legal, or technical scope.
- *Capability:* The attacker knows how and where to find redacted documents they aim to deanonymize.
- *Strategy:* The attacker extracts the redacted text from the target document and processes it through the proposed model. With the entity predictions, they can leverage other kinds of attacks (such as social engineering attacks or data leaks) or knowledge bases to fully unredacted the target document.

Practical Scenario. To accomplish the goal, the attacker of our threat model must progress through different stages: *collection*, *processing* and *recomposition*:

1. *Collection:* The attacker gains access to redacted documents. Optionally, the attacker can use other publicly available datasets similar to the target document's scope to fine-tune the unredaction system.

2. *Processing:* After being extracted, the redacted text is processed by the proposed model to gain predictions on covered entity types.
3. *Recomposition:* Combining the results of the previous step and additional knowledge (personal/public domain or combined attacks), the attacker recovers the original document's plain text.

The main focus of this paper is mainly encapsulated in the *processing* step. The *collection* step is dependent on the target that the attacker sets. Similarly, in the *recomposition* step, the attacker has a plethora of possibilities that are impossible to describe exhaustively and are, in fact, not primarily of interest to the purpose of the paper. Some essential insights may already be contained within the redacted document: one example is the length of the redacted block, which can determine with a reasonable degree of approximation how many characters the hidden text consists of.

Along the same lines, the step of extracting text from the target document (typically a PDF file) is beyond the scope of this paper. The documents are usually structured, and the text can be extracted even with a simple copy-paste from any file reader. In cases where the document is unstructured because perhaps it is the result of scanning, Optical Character Recognition (OCR) can be involved. This technology has been extensively examined by the academic community [9,22] as well as being widely adopted in the consumer world. In addition, there are OCR open source projects such as Tesseract[2] or Apache Tika[3], both widely adopted by the developers.

4 Methodology

We now detail our methodology and the techniques we use for unredaction. In Sect. 4.1, we provide the specifics of the dataset and how it generalizes real-world scenarios. Section 4.2 discusses the data processing steps applied, constituting a crucial component of our contribution. In Sect. 4.3, we detail the used models and their hyper-parameters. A complete overview of our framework is shown in Fig. 3.

4.1 Dataset

Publicly available datasets present several limitations that make their application to our study particularly challenging. In particular, most datasets are restricted to clinical records and describe only a single individual [20]. Furthermore, many are related to public figures, include only short sentences, are composed of images, or have been subject to prior redaction or anonymization techniques [30].

[2] https://github.com/tesseract-ocr/tesseract.
[3] https://tika.apache.org/.

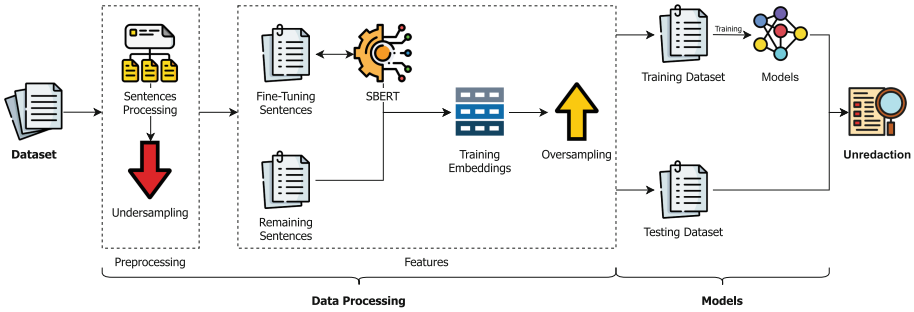

Fig. 3. RedactBuster framework overview.

Text Anonymization Benchmark. For our study, we use the TAB dataset, an open-source annotated corpus consisting of 1268 English-language court cases from the European Court of Human Rights (ECHR). Twelve university students at the University of Oslo have annotated the documents of this corpus after being trained on their classification. In particular, multiple users have annotated each document to cross-check their output. In case of disagreements, conflicts were resolved among students based on the guidelines provided by the authors. These guidelines indicate that each annotator should redact *direct identifiers* and *quasi-identifiers*. The former refers to unique values given to an individual, e.g., full names, email addresses, and social security numbers. The latter refers to publicly known information of an individual that doesn't enable identification when isolated, e.g., gender, nationality, postal code. Each document is then distributed in JSON format and contains the following data.

- Text of the court case used during the annotation.
- Document annotation.
- The target of the anonymization task.
- Whether another annotator revised the document.

Labels. The redacted text and its properties will constitute the starting point for extracting the features needed for classification. Instead, the annotations represent our study's labels and ground truth. In particular, each redacted entity belongs to one of eight classes: datetime (e.g., *"15 December 1993"*), organization (e.g., *"Court of Cassation"*), person (e.g., *"Dr Price"*), demographic (e.g., *"police officer"*), location (e.g., *"London"*), miscellaneous (e.g., *"overdose of paracetamol"*), quantity (e.g., *"2,000,000 Swedish kronor (SEK)"*), and code (e.g., *"36619/03"*).

4.2 Data Processing

When dealing with classification tasks, features and their correlation are the most significant variables models can use to compute predictions. For this reason, we first need to split the data for each annotation and define its ground truth. Then,

features must be extracted from the redacted sentences, and eventual biases in the dataset must be removed.

Preprocessing. The first step is to clean the text of each document and remove artifacts that might tamper with the classification process. In this step, we handle end-of-line characters and particular abbreviations. For instance, court texts in the dataset always include section titles (e.g., *"THE FACTS"*), which are followed by a series of `\n` characters. We handle them by substituting them with dot characters and an appropriate number of spaces since their removal would make the redaction offset wrong. We present more details of the textual-level modifications we employ in Appendix A.1. Since each document contains several redacted entities, we separate them into single samples. This facilitates classification, as ML and DL models predict one label for each sample. Therefore, we split the text into sentences using NLTK PunktSentenceTokenizer [4]. Each sentence is then associated with the calculated offset and entity type of the redactions that are present inside. After this procedure, however, we might still encounter sentences containing multiple redacted entities. For this reason, we duplicate the sentence and switch the redaction target until all confidential entities are redacted once. This also serves as a data augmentation technique, as a single sentence can represent multiple labels depending on the redacted data. An example is shown in Table 1.

Table 1. Example of sentence splitting and redaction.

Sentence	Redaction	Offset	Type
It happened on 19/10/2004.	It happened on **********.	$\langle 15, 24 \rangle$	`DATETIME`
Paolo was in Amsterdam.	***** was in Amsterdam.	$\langle 0, 4 \rangle$	`PERSON`
	Paolo was in *********.	$\langle 13, 21 \rangle$	`LOC`

At this stage, the dataset comprises the redacted text, the redaction offset, and the redaction type. However, we notice a substantial imbalance between the class distribution of the labels. Indeed, the most frequent class is datetime with 34280 samples, while code comprises only 2781 samples. This heavy skew in the data distribution can negatively impact a ML model performance, as it would create bias in classification. Therefore, we perform random undersampling on the data to balance the distribution.

Features. After the preprocessing steps, the training and testing dataset mostly contain samples in text format. However, ML and DL traditionally require a constant number of features from each sample to compute a prediction. For this reason, we use a widespread technique in the NLP literature: computing embeddings [23]. Given the importance of sentence data in our dataset, we use SentenceTransformers (SBERT), a popular Python framework for generating text and image embeddings [34]. This Python package provides several Transformer

models with different sizes and performances. We use `all-mpnet-base-v2` as it provides the best performances overall. Indeed, in our threat model scenario, resources or computational overhead do not represent an obstacle, as document unredaction is not a time-constrained task.

SBERT models are generated and trained for general purposes and do not focus on specific domains. As such, using an out-of-the-box model yields lower accuracy values with respect to a specifically trained model. For this reason, we resort to fine-tuning. With this procedure, we can retain the knowledge acquired by the model during the original training procedure and adapt it to our particular domain to increase performance. In the case of SBERT Transformers, fine-tuning is performed by comparing sentences with one another and providing a similarity score for the couple. This score is the cosine similarity. Given two vectors A and B of length n, this metric is defined as follows.

$$\cos(\theta) = \frac{A \cdot B}{||A||\,||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}. \tag{1}$$

The output of this score is then normalized on a scale from 0 to 1, where higher values indicate higher similarity between the sentences. Therefore, we extract a subset of the training dataset containing 250 samples for each label (i.e., 2000 total samples). For each label, we fine-tune the model as follows: *(i)* we group sentences in pairs (i.e., 125 for each label) and provide a similarity score of 0.8, and *(ii)* we randomly pick a sentence from that label and a sentence from another random label (until obtaining 125 couples) and provide a similarity score of 0.2. We select those values since we want our model to comprehend the similarity of the sentence context around the redaction without losing their individual properties. Indeed, since embeddings are created from redacted sentences, our Transformer model can only process the remaining sentence tokens. Therefore, phrase context provides the knowledge the classifier requires to compute predictions. It is worth noting that the 2000 samples used for fine-tuning the model are then discarded from the dataset. We do this to ensure that our Transformer model and the classifier are presented with new data and to prevent overfitting.

After fine-tuning, our Transformer generates embeddings from the remaining training sentences (i.e., 2531 for each label). The embedding size is determined by the Transformer model used for computation, which, in our case, is 768. We now oversample the data at the embedding level to compensate for the under-sampling performed during preprocessing. This is a widespread data augmentation technique used in ML literature, as it can compensate for the possible lack of data in a dataset and allow the classifier to train on more samples [15]. It is also worth noting that, in our scenario, this procedure can be applied only after the embedding computation, as oversampling textual data can provide several artifacts that can instead decrease the classifier performance. In particular, we use Synthetic Minority Over-sampling Technique (SMOTE) as oversampling technique [7]. This technique selectively generates synthetic samples through a

nearest-neighbors approach for the minority class (which, in our case, are all the classes) by interpolating between existing instances. We generate samples for each label until we obtain 3500 total samples for each class. An overview of the data balancing process is shown in Appendix A.2. We then split our dataset in training and testing respective size percentages of 85% and 15%.

4.3 Models

After dealing with the dataset and its feature extraction, we design a classifier for the unredaction task. We present a comprehensive study of different state-of-the-art models by considering three ML models and two DL models. For the ML models, we perform GridSearch on different hyper-parameters configurations to find the best values for them. In particular, we perform a 5-fold cross-validation, removing the need to include a validation set in the train and test dataset split. GridSearch is performed on a balanced subset of the original dataset (∼10000 samples). Then, the found hyper-parameters are applied to the model to train on the whole training dataset. DL architectures are instead validated manually due to the need to test multiple architectures and layers. Furthermore, being GPU-accelerated, they are significantly faster with respect to the ML executed on the CPU.

ML Models. The first model we test is the Random Forest (RF) [6], an ensemble learning method that constructs multiple Decision Tree (DT) during training. Each tree is trained on a random subset of the dataset (bagging), and a random subset of features is considered for each split, which adds randomness and reduces overfitting. Hyper-parameter search is performed on the number of estimators, the criterion, and the max depth. The first variable determines the number of trees in the forest (tested with 150, 200, and 300). The second represents the function of measuring the quality of a split (tested with gini, entropy, and log loss). The third describes the maximum depth of each DT in the forest, thus controlling the complexity of the model (tested with 3, 5, or unlimited).

Another widespread model used in the literature is Support Vector Machine (SVM) [11]. It works by finding the hyperplane that best separates classes in a high-dimensional space. For nonlinear classification tasks, SVM uses a kernel trick to map the original features into a higher-dimensional space where a separating hyperplane exists. Hyper-parameter search is performed on the kernel and the C parameter. The first variable specifies the kernel type for the algorithm (tested with polynomial and radial basis function). In the case of the polynomial kernel, we also search for its optimal degree (tested with 3 and 4). The second represents the regularization parameter that controls the trade-off between maximizing the margin and minimizing the classification error (tested with 3, 5, and 7). We also use bagging (bootstrap aggregation) to make the training process faster and reduce overfitting.

Finally, we test eXtreme Gradient Boosting (XGBoost) [8]. This model implements gradient-boosting algorithms designed for speed and performance. It

builds multiple decision trees iteratively and tries to correct the errors of the previous models. It uses a gradient descent algorithm to minimize the loss function when adding new models. Hyper-parameter search is performed on the learning rate, the L2 regularization, and the max depth. The first parameter prevents overfitting by shrinking the feature weights to make the boosting process more conservative (tested with 0.01, 0.1, and 0.3). The second variable is a regularization term on weights that penalizes the complexity of the model, preventing overfitting (tested with 0, 0.125, and 0.25). The third represents the maximum depth of each tree (tested with 3, 5, or unlimited).

DL Models. The first DL model we test is a feedforward Deep Neural Network (DNN) composed of multiple layers of neurons. Our specific implementation consists of five fully connected layers. The input layer consists of 768 neurons, corresponding to the data's input features. We include four hidden layers with 512, 256, 128, and 64 neurons, respectively. Each hidden layer applies a linear transformation followed by a Rectified Linear Unit (ReLU) activation function. Finally, the output layer consists of 8 neurons, representing the final output classes. The model is trained with cross-entropy as the loss function and uses Adam as an optimizer with a learning rate of 0.00005 for 200 epochs with a batch size of 100.

The other model architecture we use is the CNN. It is a type of neural network particularly effective for processing grid-like data, such as images or, in this case, one-dimensional sequences. CNNs utilize convolutional layers to automatically and adaptively learn spatial hierarchies of features from the input data. The reason why we select this specific architecture is for its capability to detect patterns in data samples. Indeed, in text data, embedding captures semantic information and relationships between words or tokens [37]. The convolutional layers can effectively process these to detect specific sequences at different abstract levels, such as character-level, word-level, and higher-level semantic features. In our implementation, the first two layers are convolutional. The first layer applies 16 filters of size 16 to the input sequence, producing feature maps. The second layer applies 16 filters of size 16 with a stride of 2, reducing the spatial dimensions. After each convolutional layer, a max pooling operation is applied with a kernel size of 8 and a stride of 2, which reduces the spatial dimensions while retaining significant features. Following the convolutional layers, there are four fully connected layers with 1376, 688, 344, and 172 neurons, respectively. These layers further process the extracted features. Finally, the output layer consists of 8 neurons, representing the final output classes. The model is trained with cross-entropy as the loss function and uses Adam as an optimizer with a learning rate of 0.0001 for 200 epochs with a batch size of 100.

5 Evaluation

We now evaluate the entity-type recognition capabilities of our proposed methodology. First, in Sect. 5.1, we disclose the metrics we use for the evaluation and

their definition. We then provide in Sect. 5.2 a baseline evaluation of or models in scenarios in which the Transformer model has not been fine-tuned on the corpus. We finally show the effectiveness of our fine-tuning procedure in Sect. 5.3.

5.1 Metric

From an ML perspective, our task is a multiclass classification task in which each label has the same importance. This is different from a binary classification task, where there is usually a positive and negative class. For this reason, in our scenario, it is not possible to define False Positives (FP), False Negatives (FN), and True Negatives (TN). Instead, only two events can occur when the model predicts the redacted entity type.

- *True Positive (TP)*: the predicted entity type matches the original entity type.
- *Misclassification (Err)*: the predicted entity type does not match the original entity type.

For this reason, we use accuracy as our evaluation metric, defined as the ratio of correctly classified instances (across all classes) to the total number of samples in the dataset.

$$Accuracy = \frac{\sum_{i=1}^{8} TP_i}{\sum_{i=1}^{8} (TP_i + Err_i)} = \frac{\#\text{ Correct Predictions}}{\#\text{ Total Predictions}}. \tag{2}$$

5.2 Baseline

We now present the evaluation of our models in baseline performance. This implies that each model is trained on embeddings generated from a non-finetuned Transformer model. As such, after processing the document sentences as detailed in Sect. 4.2 and undersampling them based on the number of classes, we directly compute the embeddings with the out-of-the-box model provided by SBERT. Since, in this scenario, the 2000 samples for fine-tuning the Transformer are not discarded, we reduce the oversampling percentage to reach the same number of total data samples. Training and testing datasets are then generated with the same split percentages. The results of this evaluation are shown in Table 2. DL models outperform all ML models on the test dataset, with the DNN model obtaining the best score. It is also worth noting that ML models are overfitting in most cases. This can be observed by the high scores on the training dataset, which, however, are not representative of the models' capabilities on unforeseen data. The ML models' behavior in this task can be attributed to two main factors: *(i)* the shallower architectures employed with respect to the DL models, and *(ii)* the variance and complexity of the embeddings. Indeed, the DL models reach high scores without overfitting, making their more complex architectures

advantageous in this task. Furthermore, high training scores but lower testing scores indicate that the models are memorizing the embeddings, which causes a loss of generalization capabilities. This indicates that the vectorial representation of the sentences is too statistically varied, making smaller models unable to classify them.

Table 2. Baseline evaluation of the models.

Model	Accuracy	
	Train	Test
RF	1.000	0.697
SVM	0.892	0.747
XGBoost	0.970	0.714
DNN	0.979	**0.958**
CNN	0.981	0.924

5.3 Finetuning

To address the shortcomings of the baseline evaluations, we proposed using a fine-tuned Transformer model for embedding computation. In Table 3, we provide an evaluation of our models trained and tested on the fine-tuned dataset presented in Sect. 4.2. Most of the models present an increased score in both test and train datasets with respect to the baseline of Table 2. Indeed, the best-performing model is now the CNN. Furthermore, we can also notice a significant improvement in the ML models' scores. This growth shows that fine-tuning the Transformer model makes the embedding more homogeneous, allowing for using less complex models. Regardless, the deeper architectures provided by DL should be preferred, as they can provide consistently high scores and prevent overfitting.

Table 3. Evaluation of the models on fine-tuned embeddings.

Model	Accuracy	
	Train	Test
RF	1.000	0.823
SVM	0.983	0.840
XGBoost	0.940	0.866
DNN	0.946	0.945
CNN	0.986	**0.985**

6 Countermeasures

Given the high accuracies demonstrated by our evaluation in Sect. 5, it is clear that sentence context can provide enough information for an attacker to infer the entity type of a redacted token. This threat can constitute a privacy breach since, through the entity type, data linkage can be performed, which might lead to bias and discrimination. Therefore, it is important to develop effective countermeasures depending on the field of application of the redaction. For example, if sensitive documents are shared in PDF format, an option is to disable printing functionalities, thus preventing possible attackers from detecting and extracting textual information. However, this can be counter-effective, as it also impedes legitimate parties from engaging with the document. Unfortunately, this side effect also applies to other scenarios where text is extracted through OCR systems. For instance, when redacted documents are stored in paper format, their digitalization involves scanning and extracting textual data from retrieved images. Thus, to prevent an attacker from being able to extract the text, adversarial attacks against ML-based OCR systems can be crafted [35,38]. However, this also prevents legitimate parties from obtaining the document's contents.

Character Evasion. To address the shortcomings of traditional countermeasures against document unredaction, we propose a technique we call *character evasion*. This method involves substituting specific ASCII characters at test time to fool the entire unredaction pipeline. Taking inspiration from adversarial and evasion techniques, the perturbations applied to the textual data are imperceptible to the human eye but are substantial enough to fool the ML systems [16]. Indeed, one of our study's assumptions is that the attacker can access (or generate) a dataset of redacted documents. As such, the attacker fine-tunes its Transformer model and trains the DL models to process the text automatically. Traditional evasion techniques would use the models' gradients to compute adversarial attacks at the embedding level to cause misclassification. However, the attacker owns the unredaction pipeline; thus, legitimate parties cannot access the models' parameters and architectures. Therefore, countermeasures must be applied at the document level to make the attacker misclassify redacted entity types. In this case, the best strategy is to substitute characters in the text with graphically identical ones, which, however, are processed differently by models. For example, each "a" character in the text can be swapped with the "а" character. These characters are homoglyphs, i.e., they appear visually similar but are distinct characters with different Unicode code points. In particular, the former is a Latin character (`U+0061`), while the latter is a Cyrillic character (`U+0430`). We only swap the five most common letters in the English alphabet (for which an identical Cyrillic or Armenian character is available) and show them graphically with their Unicode code in Appendix B. This technique is particularly effective in cases where human readability is necessary while preserving the content's privacy. Indeed, while humans can perfectly read the text, the unredaction attack is no longer effective. An evaluation of this countermeasure is shown in Table 4. When using character evasion, the accuracy of the models drops to values close to random

guessing. This behavior can be attributed to the Transformer model, which, by processing foreign characters at test time, cannot generate embeddings statistically close to the ones on which the DL models are trained. ML models obtain similar results, with an average accuracy of 0.238. Furthermore, we can notice that fine-tuning the Transformer does not improve the results, making character evasion a robust countermeasure against our attack. It is also worth noting that several Unicode homoglyphs exist for each Latin character. Therefore, the attacker cannot simply substitute the evaded characters one-to-one before inputting the text into the unredaction pipeline. Moreover, most homoglyphs are regular characters of different alphabets, making their detection increasingly difficult when redacting text in other languages.

Table 4. Evaluation of the models with the *character evasion* countermeasure.

Model	Accuracy	
	Baseline	Fine-tuned
DNN	0.182	0.195
CNN	0.181	0.183

7 Conclusions

With the increase in document digitalization and the consequent rise in the exchange of data, it is essential to ensure that sensitive content stays private. For this reason, several redaction techniques have been proposed and used to mask private tokens inside textual documents. With the recent discoveries on the insecurity of specific techniques such as blurring and pixelation, the complete removal of the token is often preferred as, in this way, the entity is erased from the document. However, sentence context can still leak information on the entity type that has been redacted.

Contribution. This paper presented **RedactBuster**, the first entity type extraction attack on redacted tokens. Our attack leverages state-of-the-art DL models for the processing and classification of redacted sentences. We evaluated our attack on a real-world dataset, obtaining an accuracy of 0.958. Furthermore, we proposed an effective countermeasure called *character evasion*, which can aid practitioners in defending against our discovered attack.

Future Works. For future endeavors, we aim to create a new dataset that can aid this line of research. Indeed, while our dataset included several types of documents, more entity types, and document styles can further solidify the contributions provided in this paper. This opportunity also opens up the possibility

of implementing OCR system in the experimentations, thus simulating a complete document digitalization and redaction pipeline. Access to a more realistic framework can aid us in detecting new vulnerabilities and developing more robust solutions to ensure user privacy. Finally, further experiments, such as adversarially training the Transformer with evaded characters, can enhance the applicability of our countermeasure in more varied scenarios.

A Dataset

In this Appendix section, we give more details on some of the processing steps that the text must go through before being fed to the Transformer for embedding computation.

A.1 Character Preprocessing

After handling the section titles, we substitute all `\n` characters with spaces, as some sentences might be contained in multiple lines. This procedure also ensures that the overall number of characters in the text remains the same. We also address possible abbreviations that are common in specific tokens. For example, "no." is often used for "number" (or "nos." for "numbers"). These words cause the tokenizer to end a sentence after the dot character. To fix this, we swap the dot character with an underscore. Nevertheless, the tokenization process introduces several artifacts in the text, as it often removes spaces after a dot character at the end of a sentence. This causes the sentences' offsets to be shifted with respect to the original text. We treat sentences separately and compute the redaction offsets w.r.t. to the sentence start to solve this. Finally, once we find the word to censor inside the sentence, we substitute it with the right amount of asterisk characters.

A.2 Data Balancing

To ensure that the models do not have any biases for classification, we balance the number of data samples for each class with different techniques. An overview is shown in Table 5. First, we randomly undersample data to obtain a balanced distribution. Then, we use a fixed amount of samples for each class for fine-tuning the Transformer model. Finally, with the remaining dataset, we perform oversampling.

Table 5. Number of samples for each class at different stages of the unredaction pipeline.

Class	Dataset	Undersampling	Fine-Tuning	Oversampling
DATETIME	34280	2781	2531	3500
ORG	28828	2781	2531	3500
PERSON	13393	2781	2531	3500
DEM	6325	2781	2531	3500
LOC	6224	2781	2531	3500
MISC	5169	2781	2531	3500
QUANTITY	2963	2781	2531	3500
CODE	2781	2781	2531	3500
Total	122963	22248	20248	28000

B Character Evasion

In Table 6, we show all the characters we use for our *character evasion* technique and their respective Unicode code.

Table 6. List of character evasion substitutions.

Original		Evasion	
Char	Code	Char	Code
a	`U+0061`	a	`U+0430`
e	`U+0065`	e	`U+0435`
i	`U+0069`	i	`U+0456`
n	`U+006E`	n	`U+0578`
o	`U+006F`	o	`U+043E`

C Hardware and Software Configuration

All experiments have been conducted on a workstation with the following configurations.

- **CPU**: AMD Ryzen 5 3600X.
- **RAM**: 32 GB at 3200 MT/s
- **Operating System**: Ubuntu 20.04.4 LTS.
- **Software**: Python 3.8.10.

All ML models are implemented with the `Scikit-learn` Python package, which does not natively support GPU acceleration. Instead, DL models are implemented with Pytorch 1.7.1.

References

1. Bendersky, M., et al.: Information redaction from document data. US Patent 9,734,148 (2017)
2. Bier, E., Chow, R., Gollé, P., King, T.H., Staddon, J.: The rules of redaction: identify, protect, review (and repeat). IEEE Secur. Priv. **7**(6), 46–53 (2009)
3. Biggio, B., Roli, F.: Wild patterns: ten years after the rise of adversarial machine learning. Pattern Recogn. **84**, 317–331 (2018)
4. Bird, S.: NLTK: the natural language toolkit. In: Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions, pp. 69–72 (2006)
5. Bland, M., Iyer, A., Levchenko, K.: Story beyond the eye: glyph positions break pdf text redaction. In: Proceedings on Privacy Enhancing Technologies (2023)
6. Breiman, L.: Random forests. Mach. Learn. **45**, 5–32 (2001)
7. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: synthetic minority over-sampling technique. J. Artif. Intell. Res. **16**, 321–357 (2002)
8. Chen, T., Guestrin, C.: Xgboost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794 (2016)
9. Chen, X., Jin, L., Zhu, Y., Luo, C., Wang, T.: Text recognition in the wild: a survey. ACM Comput. Surv. (CSUR) **54**(2), 1–35 (2021)
10. Chiu, J.P., Nichols, E.: Named entity recognition with bidirectional LSTM-CNNs. Trans. Assoc. Comput. Linguist. **4**, 357–370 (2016)
11. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**, 273–297 (1995)
12. Cottrille, S.C.: Selective document redaction. US Patent 7,913,167 (2011)
13. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
14. European Parliament, Council of the European Union: Regulation (EU) 2016/679 of the European Parliament and of the Council. https://data.europa.eu/eli/reg/2016/679/oj
15. Fernández, A., Garcia, S., Herrera, F., Chawla, N.V.: Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. J. Artif. Intell. Res. **61**, 863–905 (2018)
16. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
17. Hajian, S., Domingo-Ferrer, J., Monreale, A., Pedreschi, D., Giannotti, F.: Discrimination-and privacy-aware patterns. Data Min. Knowl. Disc. **29**(6), 1733–1782 (2015)
18. Hill, S., Zhou, Z., Saul, L.K., Shacham, H.: On the (in) effectiveness of mosaicing and blurring as tools for document redaction. Proc. Priv. Enhancing Technol. **2016**(4), 403–417 (2016)
19. IVASS: I principali numeri delle assicurazioni in italia (2022). https://www.ivass.it/pubblicazioni-e-statistiche/statistiche/numeri-assicurazioni/2022/Focus_I_principali_numeri_2022.pdf. Accessed 17 Apr 2024
20. Johnson, A.E., et al.: Mimic-iv, a freely accessible electronic health record dataset. Sci. Data **10**(1), 1 (2023)
21. Kelly, D.G., Foster, B.R.: Process for electronic document redaction. US Patent 8,456,654 (2013)
22. Li, M., et al.: TrOCR: transformer-based optical character recognition with pre-trained models. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, pp. 13094–13102 (2023)

23. Li, Y., Yang, T.: Word embedding for understanding natural language: a survey. In: Guide to Big Data Applications, pp. 83–104 (2018)
24. Liu, Y., et al.: Roberta: a robustly optimized BERT pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
25. Luoma, J., Pyysalo, S.: Exploring cross-sentence contexts for named entity recognition with BERT. In: Proceedings of the 28th International Conference on Computational Linguistics, pp. 904–914 (2020)
26. Mane, S.: Method and system for advanced document redaction. US Patent 11,562,134 (2023)
27. Matichuk, B., Rebstock, J., Kraft, M.: Redaction engine for electronic documents with multiple types, formats and/or categories. US Patent 10,853,570 (2020)
28. Microsoft: Presidio: Data protection and de-identification SDK (2022). https://microsoft.github.io/presidio/. Accessed 17 Apr 2024
29. Nabbosa, V., Kaar, C.: Societal and ethical issues of digitalization. In: Proceedings of the 2020 International Conference on Big Data in Management, pp. 118–124 (2020)
30. Papadopoulos, C., Pletschacher, S., Clausner, C., Antonacopoulos, A.: The impact dataset of historical document images. In: Proceedings of the 2nd International Workshop on Historical Document Imaging and Processing, pp. 123–130 (2013)
31. Petro, D.: GitHub - BishopFox/unredacter: never ever ever use pixelation as a redaction technique — github.com (2022). https://github.com/bishopfox/unredacter. Accessed 17 Apr 2024
32. Pilán, I., Lison, P., Øvrelid, L., Papadopoulou, A., Sánchez, D., Batet, M.: The text anonymization benchmark (TAB): a dedicated corpus and evaluation framework for text anonymization. Comput. Linguist. **48**(4), 1053–1101 (2022)
33. Ramos, I.S., Dickenson, M., Nair, S.: Document redaction and reconciliation. US Patent App. 16/438,439 (2020)
34. Reimers, N., Gurevych, I.: Sentence-BERT: sentence embeddings using siamese BERT-networks. arXiv preprint arXiv:1908.10084 (2019)
35. Song, C., Shmatikov, V.: Fooling OCR systems with adversarial text images. arXiv preprint arXiv:1802.05385 (2018)
36. Tikayat Ray, A., Pinon-Fischer, O.J., Mavris, D.N., White, R.T., Cole, B.F.: aeroBERT-NER: named-entity recognition for aerospace requirements engineering using BERT. In: AIAA SCITECH 2023 Forum, p. 2583 (2023)
37. Xu, H., Dong, M., Zhu, D., Kotov, A., Carcone, A.I., Naar-King, S.: Text classification with topic-based word embedding and convolutional neural networks. In: Proceedings of the 7th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics, pp. 88–97 (2016)
38. Xu, X., Chen, J., Xiao, J., Gao, L., Shen, F., Shen, H.T.: What machines see is not what they get: fooling scene text recognition models with adversarial text images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12304–12314 (2020)
39. Zhang, R., Yang, Y., Wang, W.: Research on document digitization processing technology. In: MATEC Web of Conferences, vol. 309, p. 02014. EDP Sciences (2020)
40. Zhao, X., Greenberg, J., An, Y., Hu, X.T.: Fine-tuning BERT model for materials named entity recognition. In: 2021 IEEE International Conference on Big Data (Big Data), pp. 3717–3720. IEEE (2021)