TRANSACTIONS ON DATA PRIVACY 18 (2025) 1–27

Hash the Universe: Differentially Private Text Extraction with Feature Hashing

Sam Fletcher[1], **Adam Roegiest**[2], **Alexander K. Hudek**[2]

[1]Litera, Toronto, Canada.

E-mail: sam.fletcher@litera.com; sam.pt.fletcher@gmail.com

[2]Zuva, Toronto, Canada.

Received 6 January 2023; received in revised form 7 August 2024; accepted 17 November 2024.

Abstract. Using artificial intelligence for text extraction can often require handling privacy-sensitive text. To avoid revealing confidential information, data owners and practitioners can use differential privacy, a definition of privacy with provable guarantees. In this work, we show how differential privacy can be applied to feature hashing. Feature hashing is a common technique for handling out-of-dictionary vocabulary, and for creating a lookup table to find feature weights in constant time. One of the special qualities of feature hashing is that all possible features are mapped to a discrete, finite output space. Our proposed technique takes advantage of this fact, and makes hashed feature sets Rényi-differentially private.

The technique enables data owners to privatize any model that stores the data-dependent weights in a hash table, and provides protection against inference attacks on the model output, as well as against linkage attacks directly on the model's hashed features and weights. As a case study, we show how we have implemented our technique in commercial software that enables users to train text sequence classifiers on their own documents, and share the classifiers with other users without leaking training data. Results show that even common words can be protected with $(0.06, 10^{-5})$-differential privacy, with only a 1% average reduction in Recall and no change in Precision.

Keywords. differential privacy, natural language processing, confidentiality, tokens, feature hashing, extraction

1 Introduction

The text used to train anything but the largest language models often contains only a sample of the words that could appear in the wild. To account for unexpected words, various approaches have been developed, with one approach being feature hashing [36, 47]. Feature hashing solves the vocabulary problem by taking all the features created from text (be they single words or more complicated features) and hashing them into a known, predefined hash range. This hash range acts as the maximum "dictionary size"; no matter how many unique features are created, they are all deterministically mapped to an integer within the allowable range [36, 47].

Each hash acts as the corresponding feature's index, like in a real-world dictionary, except the features are stored in numerical order rather than alphabetical order. Each hash

then points to a vector of weights that signal the importance or relevance of the underlying feature in the given NLP (natural language processing) task. The weights can be learned and updated by a machine learning algorithm, with the hashes acting as a lookup table (or "hash table") for the features. The end result is a feature set of $(hash, weights)$ pairs. When the learned model is applied to new data (usually to classify the data, extract relevant portions, or use as a prompt), the new data is featurized and hashed in the same deterministic way, and thus connected to the relevant weights in the hash table.

The data need not be text – any data that is being featurized and hashed is applicable – however for the sake of clarity we frame this paper in terms of text. Similarly, while features can be created from any aspect of the input data, we will focus the discussion on the most common piece of sensitive information in a corpus of text: the words. For any given text, we tokenize (i.e., split) it into word segments (including symbols and numbers), and use those tokens as the building blocks for any features we might want to use to train a model.

1.1 Ensuring Confidentiality

When documents contain confidential information, a privacy-preserving technique may be required before the documents can safely be used for NLP tasks. Unknown or untrustworthy users might be given access to the NLP model, either through an open-source or free product, an AI marketplace [28], or through consumer features like spell-checkers or auto-complete functionality.

While feature hashing already provides some amount of obscurity to the raw text, privacy through obscurity is no privacy at all [3, 42]. In our scenario, there are two main attack vectors:

- an attacker with access to the raw model and who knows how the tokens are featurized and hashed can guess words, hash the corresponding features, and link them to preexisting hashes in the hash table; and
- inference attacks on the model's output are unaffected by hashing, so an attacker able to use the model can input arbitrary text and observe the output any number of times, and build up an understanding of the underlying training data and distribution of weights.

In Section 3 we describe our threat model, in which the attacker has both of the above abilities.

1.2 Motivating Example

To demonstrate how these threats might manifest in real life, we provide a motivating example using our commercial software. Our software[1] [38] enables users to upload and organize documents, collaborate with users within the same organization, apply any number of 1200+ built-in sequence classification models to extract text from their documents, and annotate and train their own models using a no-code interface. Some users wish to share their custom-built models with other organizations, or with their customers. Before this can happen, the confidentiality of the underlying documents needs to be ensured. For example, organizations may have a legal mandate to protect the confidentiality of its documents, like law firms in the U.S. do.

[1] `https://kirasystems.com/` . Free academic access is available upon request.

Table 1: U.S. States appearing in the training data of a "Governing Law" model trained in our software, and then successfully extracted from frivolous sentences.

State	Occurrences in Training Data	Extracted?
Delaware	10	Yes
New York	6	Yes
Texas	4	Yes
New Jersey	1	Yes
New Mexico	0	Yes
New Hampshire	0	Yes
Alabama	0	Yes
Utah	1	
California	1	
Connecticut	1	
Illinois	1	

As a demonstration, take a real model in our software that was trained to extract paragraphs relating to the governing law jurisdiction of contracts. The trainer has annotated text in their confidential[2] documents such as:

> "This Agreement, the legal relations between the Parties and the adjudication and enforcement thereof shall be governed by and interpreted and construed in accordance with the substantive laws of the State of New York (excepting only those conflict of laws provisions which would serve to defeat the operation of New York substantive law). Any action arising under or relating to this Agreement may only be brought, if by Ubiquity in the federal courts of the United States located in the State of Connecticut, or if by Client in the federal courts of the United States located in the State of New York, and the Parties hereto hereby submit to the jurisdiction of the said courts."

If a malicious user got their hands on this model, they might try to discover confidential information about the (otherwise anonymous) trainer by inferring which jurisdictions the trainer operates in. We simulate an attack this user could perform by creating 50 frivolous documents containing a single partial sentence of the form:

> "The laws of the State of [MASK]."

where [MASK] is replaced with one of the 50 U.S. states. Table 1 presents which fragments of text the model extracts, compared to how many times the corresponding state appeared in the training data. Even when the test documents only contain one incomplete sentence, four out of eight states that appeared in the training data are extracted, while only three of the remaining 42 states not in the training data are extracted.

[2] In reality the model used for this demonstration is trained on public documents, to avoid risking the privacy of our users. No user data is included in this paper.

Table 1 is the result of a single attack, and trying other sentence fragments could narrow down the states even further. When combined with other information (such as what industry the organization operates in) it is clear how knowing where the organization operates could constitute a privacy breach.

After applying the privatization technique described in Section 4 to the above model, the model's ability to extract relevant paragraphs is largely unaffected, and none of the 50 sentence fragments are extracted.

1.3 Our Contribution

We propose a differentially-private method of training a model on any amount of text, with any number of labels. We do so by transforming a hashed feature set into a differentially private version of the feature set. We build off preliminary work presented in [18] and present new theoretical results (Section 4.5), new quantitative and qualitative experiments (Section 5) a discussion of limitations and variations of the threat model (Section 6), a running example, and two new appendices.

The proposed technique hides all the words featurized in the feature set by making them indistinguishable from all other possible words. It protects against linkage attacks when the attacker can see the hashed feature set directly and guess words, and prevents inference attacks on the output of a machine learning model built using the feature set. It is computationally efficient, independent of any specific hashing function or training function, and can be "bolted-on" after the fact to any trained model that uses a hashed feature set.

The technique also inherits all of the benefits of differential privacy [11, 12, 14]: it is mathematically guaranteeable, and immune to any amount of post-processing or auxiliary information possessed by a malicious user. Differential privacy has quickly become the state-of-the-art in privacy preservation [1, 17, 19, 32]. It defines privacy in terms of *a priori* and *a posteriori* knowledge: the inclusion of any particular data point in a data set should not markedly affect what could have been learned from the data if that data point was not included.

In order to preserve the confidentiality of each word when the words are being mapped to hashed features, our solution requires a novel approach. Unlike most differential privacy techniques, there is no aggregation we can employ when it comes to a hash table. Since they act as indexes in a lookup table, all hashes are equally "distant" from each other, and adding "noise" to a hash is the same as entirely destroying it.

Before presenting our solution to this problem, we first provide useful background information in Section 2, and a detailed description of the threat model we are operating under in Section 3. We present our technique in Section 4, then walk through a case study of the technique's real-world practicality in Section 5. Limitations and future work are discussed in Section 6. Related work can be found in Section 7, and we conclude in Section 8.

2 Background

This section briefly introduces four building blocks that will be used to construct our solution in Secton 4. The first introduces a data structure we will be using, and the next three cover differential privacy and its relevant subfields.

2.1 Feature Hashing

Feature hashing is the process of using a hash function [2] to map features to indexes. It is sometimes referred to as "the hashing trick" [36, 47] due to how it differs from traditional hashing: instead of using each hash as a key mapped to some value, the "trick" is that the hash itself is the value. This means each hash can act as an index in a table, and can be looked up in $O(1)$ time. For example, if each pair of words in a text corpus is treated as a feature, the feature "New|York" might correspond to index 3975.

Feature hashing has two main advantages: it is computationally fast and space-efficient, and more importantly for our purposes, because the table has a specified maximum size, *it maps a potentially infinite number of features into a known, bounded hash range*.

The hashing function is deterministic; a feature will always be hashed to the same value h. Hashes cannot easily be reverse-engineered back to the original feature though – the hashing process involves overwriting and shifting the bit string so many times, that reversing the process leads to many trillions of possible original features. After all, every single possible feature in the universe can be mapped to one of the limited hashes in the hash range, so "collisions" (where multiple features map to the same hash) are inevitable if the feature space is large enough. We can think of feature hashing as a form of lossy compression, where the lossiness scales with the number of collisions. In practice, the distribution of training data and features for any given problem are well-defined and limited enough that collisions are not common.

Remark 1. When using the hashing trick, the universe of possible outputs U is finite, and known. For a particular problem, the output distribution H will only use a subset of the universe, $H \subseteq U$, from which the training data x and testing data z are then drawn from. For an adequately large x and z drawn from the same distribution, we know from the Law of Large Numbers to expect minimal covariate shift; the hash table H outputted by the featurization process $g(x)$ is assumed to be close to the hash table outputted by $g(z)$:

$$g(x), g(z) \xrightarrow[|x|,|z|\to\infty]{} H$$

2.2 Differential Privacy

Differential privacy [11] is a quantifiable definition of privacy that makes strong guarantees about the risk of a privacy breach. In the paradigm of differential privacy, the data holder makes the following promise to each user:

> "You will not be affected, adversely or otherwise, by allowing your data to be used in any study or analysis, no matter what other studies, data sets, or information sources are available." [14]

It has since been adopted as the de facto privacy standard by companies like Google [1] and Apple [22], and has been applied to numerous machine learning algorithms [17, 40]. It has also recently been adopted by the U.S. Census Bureau, who used differential privacy for all data releases of the 2020 Census [20]. However, these applications of differential privacy have faced criticism, either for using unreasonable parameters that provide no real protections [9] or for harming the utility of the data too much [41]. These two concerns need to be handled carefully for an application of differential privacy to be successful.

While other privacy definitions such as k-anonymity [43] and l-diversity [31] exist, they require careful consideration of what background information malicious users could have

access to, and how that information could be combined such that the user could make inferences with confidence [43]. Differential privacy incorporates these risks in its definition [11]. For our use case, where users have access to the Internet and could know any number of things about organizations potentially entering into contracts, differential privacy is a better fit.

For our purposes, rather than protecting the privacy of individual people, we want to protect the confidentiality of each *term* (i.e., unique word; we expand on this later) in the text corpus x. We can then define differential privacy as follows:

Definition 1 (Differential privacy [11])**.** An algorithm $f(\cdot) \rightarrow M^*$ is (ϵ, δ)-differentially private if for all possible outputs in the universe $M^* \subseteq U$, for all possible adjacent corpora x and x' that differ only by all occurrences of one term:

$$Pr(f(x) \in M^*) \leq e^{\epsilon} \times Pr(f(x') \in M^*) + \delta \tag{1}$$

The variable ϵ measures the maximum multiplicative change in output probabilities, and δ measures the maximum additive change (often thought of as the failure rate of the privacy guarantee [14]). Note that Definition 1 is symmetrical for x and x'.

For example, for a value like $\epsilon \approx 0.1$, the probability of observing any particular output should not change by more than 10% when a term in x is added or removed. In essence, the removal or addition of a data point should only have a small chance of affecting a function's output (interestingly, this is similar to the concept of over-fitting, and has been formally explored in the past [15]). δ is often set to 10^{-5}, for a 1-in-100,000 chance of failure [1].

Our goal is to design an algorithm $f(M) \rightarrow M^*$ that is (ϵ, δ)-differentially private. What f and M look like is up to us, but for now M can be thought of as a trained model, and f as a custom privatization function that modifies the model.

2.3 Rényi Differential Privacy

Rényi differential privacy (RDP) [34] reframes differential privacy (DP) in terms of the Rényi divergence between two distributions, while remaining compatible with Definition 1. Compared to using Kullback–Leibler divergence to measure differential privacy [44], RDP better models the δ privacy risk, and provides an α variable that allows for a smooth trade-off between ϵ and δ. Framed in terms of text corpora, RDP is defined as follows:

Definition 2 (Rényi differential privacy [34])**.** An algorithm $f(\cdot) \rightarrow M^*$ is (α, ϵ)-Rényi differentially private if for all possible outputs in the universe $M^* \subseteq U$, for all possible adjacent corpora x and x' that differ only by all occurrences of one term:

$$D_\alpha(f(x) \| f(x')) \leq \epsilon \tag{2}$$

where D_α is the Rényi divergence of order $\alpha > 1$ between two probability distributions P and Q defined over $\mathcal{R}$:

$$D_\alpha(P \| Q) = \frac{1}{\alpha - 1} \ln \mathbb{E}_{z \sim Q} \left(\frac{P(z)}{Q(z)} \right)^{\alpha} \tag{3}$$

It has also been shown [34] that we can convert (α, ϵ)-RDP into traditional (ϵ, δ)-DP in the following way:

Lemma 3 (Converting RDP to DP [34])**.** *If $f(\cdot)$ obeys (α, ϵ)-RDP, then $f(\cdot)$ obeys (ϵ', δ)-DP for all $0 < \delta < 1$, where $\epsilon' = \epsilon + \ln(1/\delta)/(\alpha - 1)$.*

2.4 User-Level Privacy as Term-Level Privacy

Differential privacy traditionally compares two "neighboring" data sets x and x' that differ by a single data point, such that $|x| - |x'| = 1$. This treats each data point as independent. User-level privacy is a variation that takes into account that the same "user" may appear in x multiple times, and that we want to totally hide the presence of the user, rather than just one of their data points [14]. Two data sets are said to be "adjacent" to one another if they differ by *all occurrences* of a single user, such that $|x| - |x'| = k,\ k \geq 1$.

This definition of adjacency matches our scenario, in which we want to hide the presence or absence of all occurrences of each term (i.e., unique word). Since the concept of a "user" is an ill fit for our application, we instead call it *term-level privacy*. Additionally, since multiple features can be created for each of the k occurrences of a term, we define two data sets as being adjacent if they differ by all K features associated with a given term, $|x| - |x'| = K,\ K \geq k$.

Three previous works have explored providing actual user-level differential privacy against text-based linkage attacks [33, 46, 49], also known as authorship attribution attacks. At first glance this may sound similar to our work here, however there is a difference: these works focus on protecting the privacy of the people providing the text, rather than protecting the confidentiality of the text itself, where any word may breach any person's privacy. Our goal is closer to redaction in that sense, where a malicious user could know the entire document except one word. We expand on this below in Section 3 and Remark 2.

3 The Threat Model

In our threat model, the data owner has some machine learning model M they wish to make public, while leaking as little information about the data x used to train the model as possible. Model M is made up of:

- a data-dependent feature set F of hashed features H and weights Θ, and
- data-independent logic (i.e., functions, algorithms, or architecture) $\mathcal{L}$ that manipulates the feature set.

Θ can be thought of as a weight matrix, where the index of each row (i.e., vector) is equal to a hash h in H, and each column is a random variable X_i. For clarity we describe the values in Θ as "weights" as a catch-all term for any data-dependent random variables, both continuous and discrete.

A malicious user (or "attacker") may wish to uncover some number of original features generated from the training data. The attacker is assumed to have unlimited computing power at their disposal, and any amount of auxiliary information about the data, either now or in the future. Auxiliary information can include secondary sources of information such as other data repositories, information gathered via social engineering, and estimates based on real-world knowledge or personal experience. Any information that is learned about the data x from a source other than x is considered auxiliary information.

The attacker is assumed to be able to reproduce the featurization of the data and the hashing of the features (i.e., $g(x)$). They also know how privacy is added to the outputted feature set (presented in Section 4), but do not know any cryptographically-secure randomly generated numbers used when adding privacy. Note that the attacker does not possess the source code that generated the data-independent logic $\mathcal{L}$, such as the training algorithm. We explore the additional attack vector opened up by this possibility in Section 6.4.

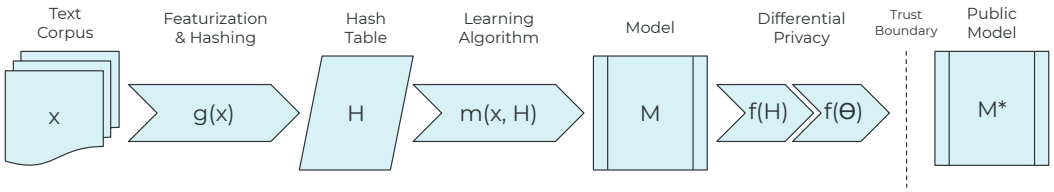

Figure 1: The process of training a model on a data set of text, then applying differential privacy on the two data-dependent components of the model (the hash table H and weight matrix Θ) to create a version of the model that preserves the privacy of all of the terms in the text.

To model the worst-case scenario, differential privacy assumes the attacker already has knowledge of all the data found in x except for one data point, and by gaining access to x, hope to discover that one data point. For example, only a single term in the training documents may be unknown to the attacker.

Remark 2. In the scenario described, we can imagine the attacker has gained a copy of some document in x from an outside source, but with one term redacted. Seeing the context surrounding the redacted word may give them some clues as to what the redacted term is. One of the advantages of the definition of differential privacy is that it incorporates these sorts of linkage or correlation attacks, as part of the attacker's *a priori* knowledge. We are measuring the increase in risk (the differential), not the absolute amount.

Our goal is to prevent the attacker from increasing their confidence about the identity of any data point. More explicitly, we want to hide the identity of the features, to prevent the attacker from discovering the raw data used to make those features. Note that the weights associated with the features are not sensitive in their own right – they are only sensitive insofar as they relate to the features. If the features are unidentifiable, any associated weights are meaningless. For example, it doesn't matter if the attacker knows that hash 3975 has weights $\{0.55, 0.14\}$ if they have no way of knowing what feature generated the hash.

Rather than perturbing the internal mechanisms of some machine learning algorithm m, we instead perturb the feature set F in the outputted model $M = \{\mathcal{L}, F\}$. This approach is known as *output perturbation* [14], and allows our solution to be "bolted-on" [48] to a variety of models, rather than being closely tied to models trained by particular algorithms. Any algorithm that builds a model from hashable features is applicable, such as Conditional Random Fields (CRF's), Hidden Markov Models (HMM's), and Support Vector Machines (SVM's).[3] Implementations of feature hashing can be found in software packages such as Tensorflow, sci-kit learn, Apache Mahout, R, Gensim, sofia-ml, Apache Spark, and Vowpal Wabbit.

Our threat model assumes that the attacker has unlimited time and access to the whole model, and so is a *non-interactive* setting [14]. This includes not just access to the input and output (such as via an API), but also to the internals of the model itself. We therefore need to not only protect against inference attacks, such as inferring the presence of a data

[3]For differentially private deep learning models, we refer the reader to previous work [1]. While popular, deep learning techniques require significantly more time and compute power, often costing 10,000 times more dollars (and CO2 emissions) to train than an equivalent CRF [10]. In our own internal testing, we have not found it to offer substantial utility improvements in our domain to merit the increased monetary, infrastructure, and environmental costs.

Table 2: A toy example of three features being hashed by $g(x)$, and then assigned up to two weights by $m(x, g(x))$.

Raw Text	**Example Feature** (2-gram)	**Hash**	**Weights before privacy** X_1	X_2
New York	$New\|York$	3975	0.55	0.14
Google Inc.	$Google\|Inc.$	3977		0.72
lung cancer	$lung\|cancer$	3980	-0.91	0.28

point based on what the model outputs when given arbitrarily specific inputs, but also against table linkage attacks, where the attacker can view the hashes and weights directly and attempt to reverse-engineer the features [19].

The concrete version of our problem setting, as it relates to textual data, is as follows. Some function $g(x)$ transforms a corpus of documents x into features that are hashed and stored in a hash table $H \subset \mathbb{N}$. Each hash $h \in H$ is an integer between 1 and a finite number R, such as 10^6.

A machine learning algorithm $m(x, H)$ is then trained on x, using H as the lookup table for storing and updating the learned weights, producing some model $M = \{\mathcal{L}, F\}$. The algorithm m might use information such as the frequency and ordering of the features found in x during training, but this information is not stored in M – the only data-dependent information in M is encoded in the feature set F. As far as privacy-preservation is concerned, we can ignore any data-independent framework or logic $\mathcal{L}$.

Each element in F is a tuple mapping hashes h to weights θ_h, $F = \{(h, \theta_h); \forall h \in H\}$, where each $\theta = \{w_i; \forall i, 0 < i \leq d\}$, and each w_i is a realized random variate from some random variable X_i among d random variables. We also use a second construction that isolates the θ vectors: weight matrix $\Theta = \{\theta_h; \forall h \in H\}$, thus allowing us to write the feature set as $F = \{H, \Theta\}$.

Some toy examples of hash tuples are included in Table 2. Note that to increase generality, we do not require every hash to have a realized value w_i for every possible X_i.

There is no correlation between small changes in a feature and the resulting hash; the hashes are approximately uniformly randomly distributed. Given that $g(x) = H$ is also a product of x, we simplify $m(x, H)$ to $m(x)$ when context allows. See Fig. 1 for a diagram of the training and privatization process (the details of the privatization function f are described in Section 4).

We can treat both g and m as black boxes for our purposes; how featurization and learning occurs does not affect our method. Our solution remains the same regardless of whether the hashes are created from individual words, n-grams of words [23], or in conjunction with other information like font or layout. Each word can be part of multiple hashes without affecting our approach, and the causal relationship between a word and multiple features is taken into account.

The attacker is assumed to have arbitrary computational power at their disposal, and an arbitrary amount of auxiliary information. For example, the attacker can have all the words in all the documents in the corpus x, except with all occurrences of one term redacted. Even in that scenario, our solution protects the confidentiality of the redacted term. Other

possible risks are:

- The attacker knows the featurization and hashing algorithm g that was used, allowing them to guess features and check if the corresponding hash appears in the hash table.
- The attacker has a template of the document that was hashed, with all the text filled in except for the spaces left for personalized information.
- The attacker has the resulting model M, and can freely input specially crafted text and observe how M's output changes, iterating as many times as they wish to see which words trigger stronger responses from M.

Our solution, presented in Section 4, protects against all these risks.

4 Term-Level Differential Privacy

Unlike most differential privacy techniques, there is no aggregation we can employ when it comes to a hash table H – all hashes are equally "distant" from each other, and adding "noise" to a hash equates to entirely destroying it. The solution needs to account for the fact that if the attacker has our hashing function, they can guess words to hash and look for them in the hash table. Our solution needs to release a hash table containing legitimate hashes, while simultaneously not allowing an attacker to detect which hashes are legitimate.

At a high level, we do this by *hashing the entire universe*. We fill in the (finite) hash table, causing legitimate hashes to become indistinguishable from the synthetically-generated hashes. Any feature that could possibly exist will have the same hash as countless other features, and will be given weights from the same distribution as every other feature, meaning that the model will act as if it saw every possible term in the training data. As long as Remark 1 holds and enough hashes correspond to the same features in new documents as they did in the training documents, the model's utility can remain largely unaffected.

Rather than perturbing x or a particular machine learning algorithm, our anonymization process $f(M) \rightarrow M^*$ uses output perturbation, modifying M after it is outputted by $m(x, H)$ but before it is publicly released.

Recall that $M = \{\mathcal{L}, F\}$ and that only the feature set F is data-dependent (and thus in need of privacy preservation). We break up F and its private version F^* into two components: $F = \{H, \Theta\}$ and $F^* = \{H^*, \Theta^*\}$ (or alternatively, $F^* = \{(h^*, \theta_h^*); \forall h^* \in H^*\}$). Similarly for the privacy function f, we can break it up into $f(H) \rightarrow H^*$, and $f(\Theta) \rightarrow \Theta^*$. We can then recombine the privatized hash table and weight matrix to make $M^* = \{\mathcal{L}, F^*\}$. Only M^* is ever made available to the public.

Fig. 1 provides a flowchart of the process. We present our strategy for $f(H)$ and $f(\Theta)$ separately, and prove that these parts are $(0, 0)$-differentially private and (ϵ, δ)-differentially private respectively in Sections 4.1 and 4.2 below.

4.1 Anonymizing the Hash Table

While trivial to prove, we include the following claim for completeness.

Claim 1 (Anonymizing H)**.** From all possible hash values 1 to R, we note the hashes that do not appear in H. The function $f(H)$ "fills in" the noted missing hashes to produce H^*, such

that $H^* = \{1, 2, \ldots, R\}$ and $|H^*| = R$. The resulting hash table H^* is (ϵ, δ)-differentially private, with $\epsilon = 0, \delta = 0$.

Proof. Observing that H is the output of $g(x)$, we can consider two adjacent corpora x and x' as in Definition 1. Function $f(H)$ always outputs H^*, no matter what x is. Thus $Pr(f(g(x)) = H^*) = 100\%$ for all possible x. Using Equation 1, we trivially have:

$$\begin{aligned} Pr(f(g(x)) = H^*) &= Pr(f(g(x')) = H^*) \\ Pr(f(g(x)) = H^*) &\leq e^\epsilon \times Pr(f(g(x')) = H^*) + \delta \\ e^\epsilon &= 1,\ \delta = 0 \\ \epsilon &= 0,\ \delta = 0 \end{aligned}$$

□

Remark 3. Not only does $f(H) = H^*$ hide which hashes were originally in H, but it also hides the original *size* of H. This prevents the attacker from knowing how many features were created from x, and also hides the collision rate of hashes in H. There is no way for the attacker to learn how many words in x were likely mapped to the same hash by only observing H^*.

4.2 Anonymizing the Weight Matrix

To anonymize Θ we need to ensure each and every hash has a full set of plausible weights. To be "plausible", the weights need to follow the distribution observed in Θ, making the synthetic and genuine hashes indistinguishable from each other. The weights themselves are meaningless to an attacker without the ability to identify the hashed features they are associated with, and so do not need privacy protection in and of themselves. To use an earlier example, it doesn't matter if the attacker knows that hash 3975 has weights $\{0.55, 0.14\}$ if they have no way of knowing what feature generated the hash.

Making synthetic and genuine hashes indistinguishable from each other includes filling in any gaps in the θ vectors corresponding to preexisting hashes $h \in H$. Depending on the machine learning algorithm m used, it may or may not be possible for some hashes to be missing some elements of θ. For example in a classification algorithm, weights might only exist for class labels the associated feature was observed to have.

To make the hashes indistinguishable, we generate synthetic weights from the same distribution as the genuine weights. Our privatization function $f(\Theta)$ first fits a d-dimensional mixture distribution $\mathbf{X} = \{X_1, \ldots, X_d\}$ to the realized random variates in $\hat{\Theta} \subseteq \Theta$, where $\hat{\Theta}$ contains only θ vectors with no missing elements. We can also write this distribution as $\mathcal{D}(\hat{\Theta})$. Each weight $w_i \in \theta \in \Theta$ can be considered to be drawn from the random variable X_i, for each $i = 1, \ldots, d$.

For each of the synthetic hashes h^* (i.e., the hashes that are not in H), $f(\Theta)$ then generates weights for all possible d elements in θ_{h^*} by sampling from $\mathbf{X}$. For preexisting hashes h, $f(\Theta)$ only generates weights from X_i for any unrealized weights w_i, conditioned on the preexisting weights in θ_h. This can be done with Cholesky decomposition [25] or a similar appropriate technique.

Both continuous and discrete random variables can be used, as well as any appropriate fitting function. If the distributions of the d random variables in Θ are non-parametric, $f(\Theta)$ can use Kernel Density Estimation (KDE) [25] or a similar technique to perform the fit [4].

Table 3: A toy segment of a filled-in feature set outputted by $f(\Theta)$, containing the example features shown in Table 2.

Before Privacy			**After Privacy**		
H	X_1	X_2	H^*	X_1	X_2
			...		
3975	0.55	0.14	3975	0.55	0.14
			3976	0.61	0.12
3977		0.72	3977	0.08	0.72
			3978	0.05	0.83
			3979	-0.33	0.49
3980	-0.91	0.28	3980	-0.91	0.28
			...		

Otherwise parametric fitting functions can be used. The better the fit is to the distribution produced by $m(x, H)$, the less erratically the distribution may change when weights are added or removed in neighbouring distributions, and the lower the privacy cost will be.

Table 3 shows a toy segment of a filled-in feature set outputted of $f(\Theta)$, continuing the example from Table 2. We provide a case study in Section 5 where we fit a multivariate mixture of Gaussian distributions to Θ, and use Cholesky decomposition [25] to generate new θ_h^* vectors and new conditional random variates $w \in \theta_h$.

4.3 Measuring the Privacy Cost

Now that $f(\Theta)$ is defined, we can measure its privacy cost using Rényi differential privacy:

Claim 2 (Anonymizing Θ)**.** Using the variables and processes described in Section 4.2, for any given term appearing in up to K features in H, function $f(\Theta)$ is (α, ϵ)-Rényi differentially private (defined by Definition 2) for all possible adjacent $\hat{\Theta}$ and $\hat{\Theta}'$ that differ by K features:

$$D_\alpha(\mathcal{D}(\hat{\Theta}) \| \mathcal{D}(\hat{\Theta}')) \leq \epsilon \ , \ s.t. \ |\hat{\Theta}| - |\hat{\Theta}'| = K \tag{4}$$

Additionally, $f(\Theta)$ is (ϵ', δ)-differentially private, where $\epsilon' = \epsilon + \ln(1/\delta)/(\alpha - 1)$ for any given $\alpha > 1$ and $0 < \delta < 1$.

Proof. The calculation of ϵ and ϵ' is a direct application of Definition 2 and Lemma 3 respectively, using the concept of adjacent data sets described in Section 2.4. The probability distribution $\mathcal{D}(\hat{\Theta})$ used in $f(\Theta)$ is defined by the subset $\hat{\Theta} \subseteq \Theta$ described in Section 4.2, and we assume the worst-case scenario where all K features are present in $\hat{\Theta}$. □

All vectors produced by $f(\Theta)$ are fitted to or generated by the distribution $\mathcal{D}(\hat{\Theta})$. The output of $f(\Theta)$ is always a weight matrix Θ^* of length R with no missing vectors θ_{h^*}, $\forall h^* \in H^*$, and no missing weights $w \in \theta_{h^*}$. For any given adjacent weight matrix Θ' known to the attacker, the only available attack vector to detect one or more of the K unknown genuine hashes is to detect θ's in Θ^* that diverge from the expected distribution $\mathcal{D}(\hat{\Theta}')$, either by guessing features and hashing them or by inferring the features by how the model behaves

on different inputs. Note that Rényi divergence also captures the effect of any outliers in the tails of the distributions, so features with unusual weights are accounted for in the privacy cost.

Note that the privacy cost ϵ is defined descriptively, not prescriptively. The data owner lacks the ability to specify ϵ, and instead must measure the cost after-the-fact and decide whether to accept the cost or modify x, $g(x)$ or $m(x, H)$ and try again. This is similar to Dwork's Propose-Test-Release framework [13], except that rather than external users querying the model interactively and consuming part of the privacy budget for each test, the data owner can test different configurations for no cost.

Remark 4. K is different for different terms. This can move adjacent distributions closer or further away, and the privacy cost ϵ will likely be higher for more frequent words, where K is large. Similarly, rarer words are more protected.

Remark 4 enables the data owner to calculate the privacy cost for specific terms if they want. This also allows for the acceptable privacy cost of extremely common terms and punctuation like "the" and "." to be higher than the acceptable cost of rarer terms. Also note that the cost of any given term is being calculated independently of the rest of x, so Parallel Composition [14] applies – the costs of each term do not add up.

The effectiveness of Claim 2 in practice depends on whether reasonable ϵ values can be achieved, and how heavily the performance of the model is affected by anonymization. We empirically demonstrate what privacy costs and performance drops can be expected in Section 5, but first we offer some insight into why we can expect performance to remain largely unaffected.

4.4 Model Utility

Remark 1 described how, when we expect the training data and any future data to be drawn from the same distribution, we can infer that the the output distribution H will also converge to the same subset of the universe of all hashes in range R. This means that we can expect it to be rare for future data to produce hashes not seen during training.

The genuine hashes $h \in H$ (and associated weights $w \in \theta_h$) remain untouched by $f(H)$ and $f(\Theta)$. Only hashes and unrealized weights that were not part of the training data are affected, and these elements are by definition outside of H. When the same h's are seen again in future data, they are correctly assigned the unperturbed weights contained in θ_h. Any weights in θ_h that were generated by $f(\Theta)$ will also be assigned, but are expected to have little impact, as they were not seen during training. The rarity of observing new hashes outside the training distribution means that the addition of the fake hashes h^* does not overly distort the predictions of the anonymized model M^*. We therefore expect model utility to remain high after making the feature set differentially private.

Of course, the sampling assumption of Remark 1 weakens as the training size decreases; a small sample $\hat{x}$ may not converge to $\mathbb{E}[x]$ as well as a larger $\hat{x}$ would.

4.5 Updating the Anonymized Model

Sometimes a data owner may wish to update a model M^* with a new batch of training data and/or more learning. So far we have only considered a single model, which we can write as $M_{b=1}$. In order to anonymize models after the first, $M_{b>1}$, we propose two changes to $f(\Theta_{b>1})$:

- $\hat{\Theta}_b \subseteq \Theta_b$ is defined as all the θ vectors that are present in the new training data or otherwise affected by the update.
- The update amounts (i.e., the element-wise differences between Θ_{b-1} and Θ_b) are considered as additional dimensions in the multivariate distribution $\mathbf{X}$ fitted to $\hat{\Theta}_b$, for a total of $2d$ dimensions.

Then $f(\Theta_b)$ can sample updates for each θ that was not updated by the new training round m_b, maintaining any correlations defined by $\mathbf{X}$. The sampled update amounts are then added to Θ_b's weights.

Claim 3 (Anonymizing $M_{b>1}$)**.** Anonymizing M_b when $b > 1$ does not require performing $f(H_b)$. $f(\Theta_b)$ is still required in order to anonymize all θ's that were not updated by m_b. For every $b > 1$, $\hat{\Theta}_b$ is the set of θ vectors that were updated by m_b. Assuming that the same g, m and α from $b = 1$ are used, and that an attacker could have access to all previous models $1 \leq i < b$, $f(\Theta_b^*)$ is $\left(\alpha, \sum_{i=1}^{b} \epsilon_i\right)$-RDP. It then follows that $f(\Theta_b^*)$ is $(\epsilon',\ \delta)$-DP, where

$$\epsilon' = \frac{\ln(1/\delta)}{(\alpha - 1)} + \sum_{i=1}^{b} \epsilon_i.$$

Proof. The function $f(H_{b>1})$ is not necessary because all hash values have already been filled in: $H_{b>1} = H_1^*$. The proof for $f(\Theta_{b>1})$ follows the same process as the proof for Claim 2, since Claim 2 holds for any number of dimensions d. We can use the composition rule described by Proposition 1 in previous work [34] to include the cost of the previous iterations, $f(\Theta_i); 1 \leq i < b$. □

Since any given term can appear in multiple updates, the cost of an attacker looking for the term needs to be paid each time. Based on the amount of risk that is considered acceptable, a cap on ϵ will limit b. Once the limit is reached, updates can be prevented in order to avoid exceeding the acceptable risk threshold. Fortunately, one of the benefits of RDP is that any applications of $f(\Theta_{b>1})$ do not increase δ when converting to (ϵ', δ)-DP, and is only added to ϵ' once.

5 Case Study

The privacy solution described in this paper is used in our commercial software, as part of a model-sharing feature where organizations can securely share their custom-built models with outside organizations. Leading up to the release of the feature we conducted two types of experiments, comparing the private models to their original, non-private counterparts. The first was a quantitative experiment, measuring the privacy cost and accuracy loss of 20 models after our privacy solution had been applied. We present our quantitative findings below in Sections 5.1.

The second experiment was conducted by our in-house team of domain experts (lawyers), who qualitatively inspected 26 private models on 1200 documents to see if there is any noticeable difference in the length and types of extractions made. An assessment of 8,182 extractions found only six noticeable changes after applying privacy. Further details can be found in Appendix B.

The documents used in our models come from the EDGAR database [6]. While these documents are public, similar agreements in the real world could easily contain highly sensitive information. For each model (described in Appendix A), in-house lawyers annotated relevant text segments in various types of contracts. The annotations, making up between 0.2% and 5.5% of the corpus for the model, are labeled as "relevant", and the remaining text are labeled as "not relevant".

Each hash tuple (h, θ) has two elements in θ, corresponding to one weight per label. Therefore the dimensionality of the fitted distribution $\mathbf{X}$ for $f(\Theta)$ is $d = 2$. Each θ_h only contains a weight for labels that $m(x, H)$ observed the corresponding h having in x.

We train Conditional Random Field models [29] using the Passive Aggressive algorithm [7] in the CRFSuite software [37] to find the relevant text. We use MurmurHash3 [2] to perform the feature hashing, with a hash range of $R = 2^{21} \approx 2 \times 10^6$.

To featurize the text, we use the punkt algorithm [27]. Our features include two that are created for each term (a uni-gram and a word vector clustering feature) and up to 10 features created from each occurrence of a term (bi-grams and 4-2-skip-grams [23]).

We find that the weights learned by the Passive-Aggressive algorithm are approximately normally distributed, and there is a high inverse correlation between the weights of the "relevant" and "not relevant" class labels. This means we can use a mixture of univariate Gaussian distributions, though not a multivariate Gaussian distribution. We use the closed-form equation [21] to calculate the Rényi divergence for each univariate Gaussian. The most divergent adjacent distribution for each dimension is created by removing the K data points furthest from the mean. The Rényi divergence to the adjacent distribution in each dimension are summed together to form the final privacy cost ϵ.

5.1 Quantitative Assessment

In this case study we focus on 20 models trained on various collections of credit, loan and lease agreements. We provide descriptions and statistics of the 20 models in Appendix A. To assess the performance of the models we train on 80% of the documents and test on the remaining 20%. We use Recall, Precision and F_1 scores [45], given the high label imbalance.

Table 4 presents the performance of the original, non-private models compared to their private counterparts. The cost ϵ' of privatizing each model when $\delta = 10^{-5}$ is listed in the final column. We chose this value of δ for this case study as it roughly matches the number of terms used to train each model (see Table 5). A larger value could have been used, but at a cost to ϵ, and we wanted to strike a balanced level of privacy protection.

On average, there is no loss in Precision or F_1 scores [45], and only a .01 reduction in Recall. In six cases, F_1 scores actually increase by $.01 - .02$ points. These improvements in performance are likely due to the model benefiting from the regularizing effect of differential privacy [15]. Aside from Model *(r)* (which did not achieve high accuracy even without privacy being added), the F_1 scores reduce by $.01 - .02$ points in five cases. Interestingly, Model *(r)*, the worst-performing model, experiences a .10 reduction in F_1 score, suggesting that problems with poorly-fitting weights are exacerbated when filling in the rest of the hash table with that same distribution of weights.

As noted in Remark 4, K can be calculated separately for any given term when calculating the privacy cost. Fig. 2 shows the privacy cost ϵ' of terms that appear in different percentages of the total number of features when $\delta = 10^{-5}$. Zipf's Law [50] gives us an approximation for the relevant frequency of common terms. For example, "the" is the most common term, and appears in approximately 7% of all features. Zipf's Law tells us that less frequent terms appear inversely proportionally to their frequency rank, so for example

Table 4: The Precision, Recall, and F1 scores of 20 (ϵ', δ)-differentially private models, compared to their original non-private versions. The privacy cost ϵ' of the 100th most common word in each corpus when $\delta = 10^{-5}$ is also reported.

Model	**Original Model**			**Private Model**			ϵ'
	Precision	Recall	F1	Precision	Recall	F1	
a)	1.00	0.83	0.91	1.00	0.83	0.91	0.048
b)	1.00	0.86	0.92	1.00	0.86	0.92	0.034
c)	0.88	0.90	0.89	0.90	0.90	0.90	0.063
d)	0.95	0.95	0.95	0.96	0.95	0.96	0.076
e)	0.70	0.95	0.80	0.70	0.95	0.80	0.057
f)	0.72	0.97	0.83	0.74	0.97	0.84	0.066
g)	0.91	0.91	0.91	0.91	0.89	0.90	0.086
h)	0.72	0.95	0.82	0.74	0.95	0.83	0.143
i)	0.92	0.87	0.89	0.93	0.84	0.88	0.100
j)	0.81	0.93	0.87	0.82	0.92	0.87	0.092
k)	0.86	0.84	0.85	0.86	0.84	0.85	0.051
l)	0.96	0.89	0.93	0.93	0.89	0.91	0.056
m)	0.92	0.85	0.88	0.92	0.85	0.88	0.080
n)	0.97	0.93	0.95	0.97	0.93	0.95	0.032
o)	0.93	0.69	0.79	0.96	0.69	0.81	0.048
p)	0.92	0.94	0.93	0.89	0.94	0.92	0.038
q)	0.97	0.77	0.86	0.97	0.74	0.84	0.076
r)	0.48	0.69	0.56	0.39	0.56	0.46	0.031
s)	0.94	1.00	0.97	0.94	1.00	0.97	0.042
t)	0.95	1.00	0.98	0.98	1.00	0.99	0.049
Average	**0.88**	**0.89**	**0.87**	**0.88**	**0.88**	**0.87**	**0.063**

the 100th most common term appears in $K \approx 0.07\%$ features. Sensitive words, such as company names, are likely to be far less common than this, and we can see in Fig. 2 that the privacy cost of the 1000th most common word is $\epsilon' = 0.016$.

Dwork, the creator of differential privacy, has recommended that the privacy cost remain at or below 0.1 [12], however state-of-the-art machine learning algorithms often go as high as $\epsilon = 1$ [17], $\epsilon = 2, 4, 8$ [1] or even $\epsilon = 8.6$ [32]. We find it promising that our technique can provide guarantees as strong as $(0.063, 10^{-5})$-differential privacy for even reasonably common terms, with only minor degradation in model performance.

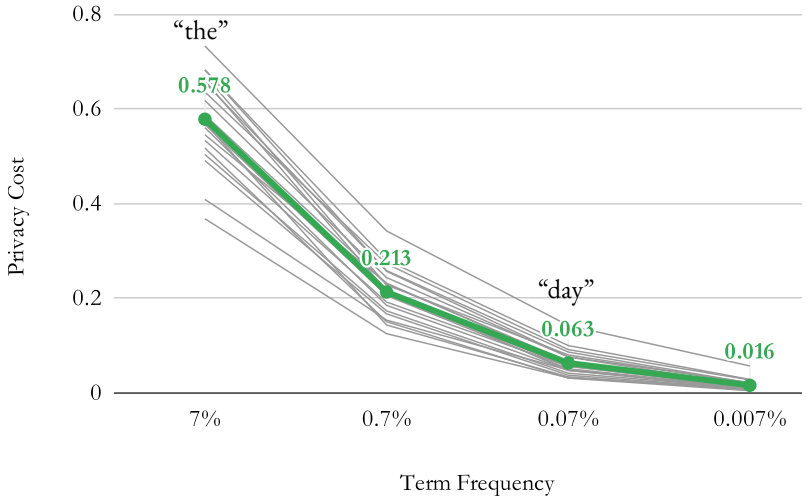

Figure 2: The privacy cost ϵ' of terms at four different frequencies. The thick green line shows the mean of the privacy costs for the 20 models (grey lines) described in Appendix A. The first and third frequencies are labeled "the" and "day" as examples of terms that occur at approximately that frequency. The privacy costs for all models when $K = 0.07\%$ can be found in Table 4.

6 Limitations and Future Work

This work represents the first attempt at using differential privacy to hide the hashes present in a hash table. This scenario exists in stark contrast to more traditional applications of differential privacy, where noise is added to data such as counts, linear queries, and summary information. Given this novelty, we believe there are multiple areas where improvements to our technique can be explored. At the highest level, there are likely many more applications where our key observation – that finite output spaces can be filled in – can lead to differentially private solutions. For this initial work, we list several specific areas that could likely be improved.

6.1 Composing the Risk of Correlated Unknown Terms

Remark 2 describes an attacker having a copy of a document except for one missing term. For example, a form with one fillable blank space. If there are multiple unknown terms, and the attacker manages to increase their confidence in one of them, other correlated terms could become proportionally more at risk of exposure. Rényi Sequential Composition [34] enables us to sum together these risks in the same way as described in Claim 3: the ϵ costs are summed and δ remains constant.

In practice, the impact of sequential composition depends on the use case, and the number of correlated unknown terms. For example, if a standard form has ten pieces of personal information filled in, we can increase the term frequency by an order of magnitude and estimate the risk in our case study using Figure 2. Alternatively, the document may be bespoke, with hundreds of correlated terms. If the combined frequency of these terms was similar to the frequency of "the", the risk in our case study would rise to $\epsilon = 0.578$.

6.2 Efficiently Measuring the Privacy Cost

When fitting $\hat{\Theta}$ to a distribution $\mathbf{X}$, it may not be feasible to use a multivariate parametric distribution, such as a Gaussian distribution. While techniques such as KDE are still likely suitable [4], the resulting distribution will lack a closed-form solution for measuring the Rényi divergence to adjacent distributions [21]. If this is the case, unless the features associated with each word are explicitly known, it may be necessary to brute-force build every adjacent distribution to find the one furthest from $\mathbf{X}$. For K features, this may require $\binom{|\hat{\Theta}|}{K}$ distributions, with a complexity of $\mathcal{O}\left(\min\left(|\hat{\Theta}|^K, |\hat{\Theta}|^{|\hat{\Theta}|-K}\right)\right)$ excluding the complexity of fitting each of those non-parametric distributions.

If each random variable X_i behaves parametrically in isolation, one alternative is to fit each one to univariate parametric distributions separately, with the resulting multivariate distribution being a mixture distribution. In this scenario, we can measure the closed-form Rényi divergence [21] of each distribution separately, and then add up the respective privacy costs. This gives a naive upper bound on the privacy cost – it is likely that tighter bounds exist, especially if one realized random variate can be used to derive the remaining random variables.

6.3 Computational Complexity and Storage Requirements

The computational complexity of $f(H)$ is trivially $\mathcal{O}(1)$, while $f(\Theta)$'s complexity largely depends on whether non-parametric fitting techniques like KDE are required. Fortunately, even in the case of non-parametric techniques, $f(\Theta)$ scales with $\hat{\Theta}$ and not x, making it likely substantially faster than the training process $m(x)$. This is ideal given the bolt-on nature of our proposed technique, where it is applied once to a previously-trained model. Additionally, due to the nature of feature hashing, using H^* on new data remains at $\mathcal{O}(1)$ as H^* is still a lookup table.

The trade-off is that the storage requirements of M^* are likely much larger than M's. This is due to both the hash range H and the weights $w \in \theta \in \Theta$ being completely "filled in", for a total of $R \times (d+1)$ data points. One upside is that the storage requirements are more predictable and consistent, but depending on the size of R and d, it is possible for orders of magnitude more storage space to be required. Perhaps future work can find a way to reduce this (beyond using problem-agnostic compression techniques), but high storage requirements may simply be the cost for having differentially private feature sets. There is no such thing as a free lunch, after all [26].

6.4 Changing the Threat Model

No matter what the scenario, the privacy guarantees depend on the threat model – the abilities and limitations the attacker is assumed to have. Here we consider two variations on the threat model we presented in Section 3.

6.4.1 A lateral change

In the base threat model, we limited the scope to a single model M being shared, but assumed that M could be "cracked open" and the hash table could be directly observed. We can flip these criteria, and instead imagine the attacker's access to M being strictly controlled by an API (where they can only change the inputs and observe the outputs), but

can create their own models M' using the same training algorithm. For example in an AI marketplace, the attacker might have the ability to create their own models, either to share or for their own (perhaps nefarious) purposes. If the attacker possesses all of the training data used to train M except for one term, they would have the ability to train their own version of the same model, using a placeholder term for the missing term. This would give them a model with the exact same distribution of weights as M. Fortunately, this is the same as what they could achieve by querying the original model an unbounded number of times; learning how the distribution of M differs from an adjacent model M'. This is exactly what Claim 2 is measuring, and so the same privacy guarantees apply.

6.4.2 A stronger attacker

Things change if the attacker has *both* the ability to train new models and see the hash tables of each model directly. In this scenario, the attacker can directly observe the weights associated with the placeholder term, and know that those weights are identical to the weights of the unknown term in M. Preventing this threat is difficult to make tractable guarantees about, and would be a good direction for future work.

One possibility is to make the training process m non-deterministic in a way that can result in the weights changing drastically, such as by shuffling the training documents. Unfortunately, even if non-deterministic operations can prevent the output of m from being the same every time, an attacker could still theoretically simulate every possible run of m and search for a Θ' that matches the majority of the weights seen in Θ^* for the hashes known to the attacker. For strict differential privacy, this is assumed to be possible. Other definitions have relaxed this assumption, such as computational differential privacy (CDP) [35]. In CDP, privacy is only guaranteed against "efficient" (computationally-bounded) attackers.

Given that even a simple shuffling procedure over x can result in $|x|!$ possible input sequences, we conjecture that for non-deterministic training algorithms m, $f(\Theta)$ is ϵ-CDP for acceptably-small values of ϵ. Proving that this is the case is outside our scope, and we leave it as future work.

7 Related Work

7.1 Redaction

Redaction is currently the most common method used in day-to-day operations to maintain the confidentiality of words [8, 16, 39] or documents [24]. While some work has been done on automatic redaction [39], semi-automatic redaction [8], and human-assistance tools [16], there is often a high cost associated with failing to redact something sensitive (i.e., a false negative), making automatic redaction difficult to trust in real-world scenarios.

Unlike differential privacy, redaction also does not protect against inference attacks, where an attacker might be able to infer a redacted word by the surrounding context or by using auxiliary sources of information. For example, a company name might be redacted, but if other data points are not redacted (such as operating region or revenue) the attacker can use those data points to narrow down the possible companies. This sort of inference attack was famously seen when journalists were able to uncover the identity of users in a dataset released by AOL [3], and seen again recently when Netflix released redacted data, but their users' privacy was still breached [42].

7.2 Word Vector DP

Recent work attempted to apply differential privacy to text representations [30] in a deep learning framework. The authors take word vectors and convert the real numbers into 10-bit representations split into a sign bit, 4 bits for the integer component, and 5 bits for the fraction component. They then use a one-hot encoding technique to flip each bit with some probability, with different probabilities for even versus odd bits, and for bits set to 0 versus 1. After evaluating the paper and following up with private correspondence, we are not convinced that this approach is sound.

One-hot encoding techniques assume that each bit position is arbitrary, and this assumption is broken when using a schema like the one described above [30]. For example, flipping the sign bit has a substantially bigger impact on the resulting word vector than flipping the last fraction bit does. Moreover, due to the nature of their perturbation (where substantially more noise is added to bits at odd indexes than at even indexes, and to 0 bits than to 1 bits), it leads to some word vectors being almost completely untouched by the noise. An attacker could be confident that word vectors comprising of certain bits at certain indexes still match the vector of the original word, destroying confidentiality.

7.3 Empirical DP

Concurrent unpublished work [4] has proposed a new "empirical" form of differential privacy (not to be confused with an older technique that misused the same name [5]). The authors propose a framework that is similar to ours but with a focus on tabular data, in which the probability distribution of a dataset is compared to all possible neighbouring distributions, effectively measuring the empirical impact any one data point can have on the dataset, without any noise needing to be added.

8 Conclusion

When models are trained on confidential text, the owners of the text may want to know that none of the terms in the text will be discoverable. Differential privacy allows us to quantify the risk the terms are exposed to, and guarantee that no matter how much auxiliary information an attacker might have (now or in the future), that risk cannot increase.

We have demonstrated that by taking advantage of the discrete, finite output space used by feature hashing, it is possible to preserve the confidentiality of individual terms without having to perform any aggregation or noise addition on the genuine hashes. Instead, differential privacy can be achieved by filling the remaining hash space with synthetic hashes that are indistinguishable from genuine hashes. We have proven the privacy guarantees, and empirically demonstrated that it is possible to produce models that experience little degradation in performance with only a small privacy cost.

References

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep Learning with Differential Privacy. In *23rd ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. ACM, 2016.

[2] A. Appleby. MurmurHash3, 2012. URL https://github.com/aappleby/smhasher.

[3] M. Barbaro and T. Zeller Jr. A face is exposed for AOL searcher no. 4417749, aug 2006.

[4] P. Burchard and A. Daoud. Empirical Differential Privacy. *arXiv*, 1910.12820:1–19, 2021.

[5] A.-S. Charest and Y. Hou. On the Meaning and Limits of Empirical Differential Privacy. *Journal of Privacy and Confidentiality*, 7(3):53–66, 2017.

[6] U. S. Commission and Exchange. Electronic Data Gathering, Analysis, and Retrieval system. *SEC Docket*, 118(19), 2013. URL https://www.sec.gov/edgar.shtml.

[7] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

[8] C. Cumby and R. Ghani. A Machine Learning Based System for Semi-Automatically Redacting Documents. In *23rd Conference on Innovative Applications of Artificial Intelligence*, pages 1628–1635, San Francisco, USA, 2011. AAAI.

[9] J. Domingo-Ferrer, D. Sánchez, and A. Blanco-Justicia. The limits of differential privacy (and its misuse in data release and machine learning). *Commun. ACM*, 64(7): 33–35, June 2021.

[10] J. Donnelly and A. Roegiest. The Utility of Context When Extracting Entities from Legal Documents. In *29th International Conference on Information and Knowledge Management*, pages 2397–2404. ACM, 2020.

[11] C. Dwork. Differential Privacy. In *Automata, Languages and Programming*, volume 4052, pages 1–12, Venice, Italy, 2006. Springer.

[12] C. Dwork. Differential Privacy: A survey of results. In *Theory and Applications of Models of Computation*, pages 1–19, Xi'an, China, 2008. Springer.

[13] C. Dwork and J. Lei. Differential privacy and robust statistics. In *ACM Symposium on Theory of Computing*, pages 371–380. ACM, 2009.

[14] C. Dwork and A. Roth. *The Algorithmic Foundations of Differential Privacy*. Now Publishers, 2013.

[15] C. Dwork, V. Feldman, M. Hardt, T. Pitassi, O. Reingold, and A. Roth. Generalization in Adaptive Data Analysis and Holdout Reuse. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NIPS 2015)*, volume 28, pages 2350–2358. Curran Associates, Inc., 2015.

[16] P. E. Engelstad, H. Hammer, K. W. Kongsgard, A. Yazidi, N. A. Nordbotten, and A. Bai. Automatic Security Classification with Lasso. In *International Workshop on Information Security Applications*, pages 399–410, Jeju Island, Korea, 2015. Springer-Verlag New York.

[17] S. Fletcher and M. Z. Islam. Differentially private random decision forests using smooth sensitivity. *Expert Systems with Applications*, 78:16–31, 2017.

[18] S. Fletcher, A. Roegiest, and A. K. Hudek. Towards protecting sensitive text with differential privacy. In *Trust, Security and Privacy in Computing and Communications*, page 8. IEEE, 10 2021.

[19] B. Fung, K. Wang, R. Chen, and P. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys*, 42(4):1–53, 2010.

[20] S. L. Garfinkel, J. M. Abowd, and S. Powazek. Issues encountered deploying differential privacy. In *2018 Workshop on Privacy in the Electronic Society*, pages 133–137, Toronto, Canada, 2018. ACM.

[21] M. Gil, F. Alajaji, and T. Linder. Rényi divergence measures for commonly used univariate continuous distributions. *Information Sciences*, 249(905):124–131, 2013.

[22] A. Greenberg. Apple's 'Differential Privacy' is about collecting your data - but not your data, 2016. URL `https://www.wired.com/2016/06/apples-differential-privacy-collecting-data/`.

[23] D. Guthrie, B. Allison, W. Liu, L. Guthrie, and Y. Wilks. A closer look at skip-gram modelling. In *5th International Conference on Language Resources and Evaluation*, pages 1222–1225, Genoa, Italy, 2006. European Language Resources Association.

[24] H. Hammer, K. W. Kongsgard, A. Bai, A. Yazidi, N. A. Nordbotten, and P. E. Engelstad. Automatic security classification by machine learning for cross-domain information exchange. In *IEEE Military Communications Conference*, page 6, Tampa, USA, 2015. IEEE.

[25] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer-Verlag New York, 2 edition, 2009.

[26] D. Kifer and A. Machanavajjhala. No free lunch in data privacy. In *2011 International Conference on Management of Data - SIGMOD '11*, page 193. ACM, 2011.

[27] T. Kiss and J. Strunk. Unsupervised Multilingual Sentence Boundary Detection. *Computational Linguistics*, 32(4):485–525, 2006.

[28] A. Kumar, B. Finley, T. Braud, S. Tarkoma, and P. Hui. Marketplace for AI Models. *arXiv preprint cs.CY*, 2003.01593:1–8, 2020.

[29] J. Lafferty, A. McCallum, and F. C. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *18th International Conference on Machine Learning*, pages 282–289. Morgan Kaufmann Publishers, 2001.

[30] L. Lyu, Y. Li, X. He, and T. Xiao. Towards Differentially Private Text Representations. In *43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1813–1816, Virtual Event, China, 2020. ACM.

[31] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data*, 1 (1):3, 2007.

[32] A. Machanavajjhala, D. Kifer, J. Abowd, J. Gehrke, and L. Vilhuber. Privacy: Theory meets Practice on the Map. In *24th International Conference on Data Engineering*, pages 277–286. IEEE, 2008.

[33] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang. Learning Differentially Private Recurrent Language Models. In *Sixth International Conference on Learning Representations*, pages 1–14, Vancouver, Canada, 2018.

[34] I. Mironov. Rényi Differential Privacy. In *30th IEEE Computer Security Foundations Symposium*, pages 263–275, Santa Barbara, USA, 2017. IEEE.

[35] I. Mironov, O. Pandey, O. Reingold, and S. Vadhan. Computational differential privacy. *Lecture Notes in Computer Science*, 5677:126–142, 2009.

[36] J. Moody. Fast Learning in Multi-Resolution Hierarchies. *Advances in Neural Information Processing Systems*, 1:29–39, 1989.

[37] N. Okazaki. CRFsuite: a fast implementation of Conditional Random Fields (CRFs), 2007. URL `http://www.chokkan.org/software/crfsuite/`.

[38] A. Roegiest, A. K. Hudek, and A. McNulty. A Dataset and an Examination of Identifying Passages for Due Diligence. In *41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 465–474, Ann Arbor, MI, USA, 2018. ACM.

[39] D. Sanchez, M. Batet, and A. Viejo. Detecting Sensitive Information from Textual Documents: An Information-Theoretic Approach. In *International Conference on Modeling Decisions for Artificial Intelligence*, pages 173–184, Catalonia, Spain, 2012. Springer.

[40] A. D. Sarwate and K. Chaudhuri. Signal Processing and Machine Learning with Differential Privacy. *IEEE Signal Process Magazine*, 30(5):86–94, 2013.

[41] M. Schneider. Census Bureau tables controversial privacy tool for survey, 2022. URL `https://apnews.com/article/technology-government-and-politics-privacy-8ab4f67fbec19fd57365bd58ffde0e4d`.

[42] R. Singel. Netflix cancels recommendation contest after privacy lawsuit, 2010. URL `https://www.wired.com/2010/03/netflix-cancels-contest/`.

[43] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.

[44] S. Vadhan. *The Complexity of Differential Privacy*. Harvard University, 2017.

[45] C. van Rijsbergen. *Information Retrieval*. Butterworth, 1979.

[46] B. Weggenmann and F. Kerschbaum. SynTF: Synthetic and differentially private term frequency vectors for privacy-preserving text mining. In *41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 305–314. ACM, 2018.

[47] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature Hashing for Large Scale Multitask Learning. In *26th International Conference on Machine Learning*, pages 1113–1120, Montreal, Canada, 2009. ACM.

[48] X. Wu, F. Li, A. Kumar, K. Chaudhuri, S. Jha, and J. F. Naughton. Bolt-on Differential Privacy for Scalable Stochastic Gradient Descent-based Analytics. In *ACM International Conference on Management of Data (SIGMOD 2017)*, pages 1307–1322, Chicago, USA, 2017. ACM.

[49] J. Zhang, J. Sun, R. Zhang, Y. Zhang, and X. Hu. Privacy-Preserving Social Media Data Outsourcing. In *IEEE Conference on Computer Communications*, pages 1106–1114, Honolulu, USA, 2018. IEEE.

[50] G. K. Zipf. *Human behaviour and the principle of least effort*. Addison-Wesley Press, 1949.

A Model Descriptions

Here we provide descriptions of the 20 models used in our case study, labeled *a)* to *t)*. Accompanying statistics are provided in Table 5. For each model, our in-house team of legal experts annotated (i.e., labeled) any sentences in documents pertinent to the particular topic the model is aiming to extract. They then trained the models on the documents using our no-code training interface, iterating on their annotations based on the system's feedback until a high level of quality was reached [38].

a) Evidence of Loans This model captures the requirement of the lender to maintain records evidencing the indebtedness. This model was trained on credit, facility and loan agreements.

b) "All-In Yield" Definition This model captures the definition of "All-in Yield" or other terms defining the yield payable to lenders on loans. This model was trained on credit, facility and loan agreements.

c) "Applicable Margin" Definition This model captures the definitions of "Applicable Margin", "Applicable Rate", "Margin" or similar terms defining the margin payable on a loan. This model was trained on credit, facility and loan agreements.

d) "Base Rate" Definition This model captures the definitions of any base rates applicable to a loan, including prime rates, LIBOR rates, eurodollar rates, screen rates, interpolated rates and federal funds rates. This model was trained on credit, facility and loan agreements.

e) "Cash Equivalents" Definition This model captures the definitions of "Cash Equivalents" or "Cash Equivalent Investments" as typically referenced in a borrower's covenants. This model was trained on credit, facility and loan agreements.

f) "Collateral" / "Transaction Security" Definition This model captures the definitions of "Collateral" or "Transaction Security" provided in connection with a secured loan. This model was trained on credit, facility and loan agreements.

g) "Collateral Documents" / "Security Documents" Definition This model captures the list of documents that must be provided in connection with a grant of a security interest in collateral. This model was trained on credit, facility and loan agreements.

h) "EBITDA" Definition This model captures various definitions related to the calculation of earnings before interest, tax and amortization. This model was trained on credit, facility and loan agreements.

i) Dispositions or Asset Sales Covenant This model captures covenants of a borrower not to dispose of assets other than in the ordinary course, and will also capture the definition of "Permitted Dispositions" or any exceptions to the definition of "Asset Sale". This model was trained on credit, facility and loan agreements.

j) Financial Statements and Information Reporting Covenant This model captures covenants of a borrower to deliver financial statements and other information to the lenders or agents. This model was trained on credit, facility and loan agreements

k) Change of Control – Credit Agreement This model captures mandatory prepayments and events of default triggered by a change of control. This model does not capture covenants not to make divestitures or to undergo fundamental changes (as these concepts can be captured with separate models). This model was trained on credit, facility and loan agreements.

l) "Specified Representations" / "Repeating Representations" Definition This model captures the definitions of "Specified Representations", "Repeating Representations" and "Major Representations". This model was trained on credit, facility and loan agreements.

m) Full Disclosure / No Misleading Information Representation This model captures representations by a borrower that all factual information provided by it to the lenders or agents is true and complete in all material respects. This model was trained on credit, facility and loan agreements.

n) Assignment Transfer Fees This model captures any transfer fees payable to the agent in connection with the assignment or transfer of a loan. This model was trained on credit, facility and loan agreements.

o) Eligible Assignees This model captures the types of parties to which a lender may assign a loan. This model was trained on credit, facility and loan agreements.

p) "Approved Fund" / "Related Fund" Definition This model captures the definitions of "Approved Fund" or "Related Fund". This model was trained on credit, facility and loan agreements.

q) Costs and Expenses This model captures the requirement that the borrower pay costs and expenses associated with the loan transaction. This model was trained on credit, facility and loan agreements

r) "Excess Availability" Definition This model captures the definitions of "Excess Availability", "Availability" and similar concepts setting out the amount available to be borrowed under an asset based loan. This model was trained on credit and loan agreements.

Table 5: Details of the 20 models used in the quantitative assessment of our case study. Recall that "unique word" is the same thing as "term" in this work.

Model	Document Count	Word Count (millions)	Unique Word Count (millions)
a)	79	8	0.053
b)	194	20	0.108
c)	195	24	0.135
d)	321	30	0.140
e)	301	24	0.134
f)	290	19	0.120
g)	250	18	0.118
h)	348	28	0.140
i)	288	20	0.120
j)	365	28	0.150
k)	125	20	0.071
l)	196	15	0.083
m)	144	14	0.098
n)	365	20	0.122
o)	365	20	0.122
p)	373	21	0.128
q)	173	9	0.061
r)	300	24	0.124
s)	188	17	0.084
t)	326	25	0.142
Average	**259**	**20**	**0.115**

s) Equity Cure Rights This model captures rights of a borrower to cure a breach of the financial covenants with an equity injection. This model was trained on credit and loan agreements.

t) "FATCA" Definition This model captures the definition of "FATCA". This model was trained on credit, facility and loan agreements.

B Qualitative Assessment

For the qualitative assessment, our in-house domain experts simulated a "Company A" sharing 26 models with a "Company B". They then compared 8,182 segments of text extracted from the same 600 documents using both the original ("Model A") and shared (i.e., privatized, "Model B") models. First a script was used to remove all extractions that were identical for both Company A and B, and then the remaining extractions were manually

assessed. The models covered the following use-cases: Credit Agreements, IP and Licensing, M&A (Mergers and Acquisitions), Leases, UCC (Uniform Commercial Code) and Bond Indentures.

Out of the 8,182 comparisons, only six were different and are quoted below. Of the six differences, one was considered a "major violation" by our domain experts, where Company B's privatized version of the model missed text in an extraction, and the text would be relevant and important to the user. The other five differences were deemed either minor differences, or arguably an improvement for the private model (due to the regularization effect of even previously-unseen features having weights). These results support the findings of our quantitative assessment: the quality of privatized models remains very high.

IP and Licensing models

> "Model B captured an additional text extraction that Model A did not capture. The additional text captured relates to the purpose of the [the model in question]."

> "Model B captured an additional text extraction that Model A did not capture. The additional text captured relates to the purpose of the License Grant model (though not completely irrelevant, the text is not correctly capturing the purpose of the Exclusivity model [the model in question])."

> "Model B captured additional text in an extraction that Model A did not capture. The additional text captured relates to the purpose of the [model in question]."

> "Model B did not break the highlight (extraction) like Model A did. So, Model B performed better on this extraction."

UCC models

> "Model B missed text in an extraction that Model A correctly captured. Model B missed a line of text that it should have captured. Of all the differences described herein, this miss by Model B was the most troubling, though it missed a single line and not the entire extraction."

Lease models

> "Model B captured an additional text extraction that Model A did not capture. Model B captured language for the Base Rent model [the model in question] on an equipment lease. The Base Rent model was not trained on equipment leases, but was trained on commercial leases. The language in this equipment lease appears as if it could be a Base Rent provision, but should not have been extracted in this document. Model A performed correctly and Model B did not."