# Towards Protecting Sensitive Text with Differential Privacy

Sam Fletcher, Adam Roegiest and Alexander K. Hudek
*Kira Systems*
*Toronto, Canada*
*Email: sam.fletcher@kirasystems.com*

***Abstract*—Natural language processing can often require handling privacy-sensitive text. To avoid revealing confidential information, data owners and practitioners can use differential privacy, which provides a mathematically guaranteeable definition of privacy preservation. In this work, we explore the possibility of applying differential privacy 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. Traditionally, differential privacy involves adding noise to hide the true value of data points. We show that due to the finite nature of the output space when using feature hashing, a noiseless approach is also theoretically sound. This approach opens up the possibility of applying strong differential privacy protections to NLP models trained with feature hashing. Preliminary experiments show that even common words can be protected with $(0.04, 10^{-5})-$differential privacy, with only a minor reduction in model utility.**

## 1. Introduction

When training machine learning algorithms for NLP tasks, the training data often has a vocabulary that contains only a sample of all the words that could appear in the target population. Accordingly, techniques have arisen that account for words that were not present in the training data. One such technique is feature hashing [1], [2]. Feature hashing solves the vocabulary problem by taking all of the features created from the text – be they single words (uni-grams) 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 a hash value within the allowable range [1], [2].

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 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 or extract relevant portions), the new data is featurized and hashed in the same deterministic way, and thus connected to the relevant weights.

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. That is, we tokenize (i.e., split) the text into word segments (including symbols and numbers), and use those tokens to build the features.

### 1.1. Ensuring Confidentiality

If the documents in a corpus contain sensitive information, a privacy-preserving (i.e., confidentiality-preserving) technique may be required before the documents can safely be used for NLP tasks. Privacy-preservation may also be required in order to safely share trained models on an AI marketplace [3].

While feature hashing already provides some amount of obscurity to the raw text, privacy through obscurity is no privacy at all [4], [5]. In this case, there are two main attack vectors:

- an attacker 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 with a copy of the model can input arbitrary text and observe the output any number of times, and build up an understanding of the underlying text and the distribution of the weights.

We assume an attacker possesses both of these abilities in our threat model, described in detail in Section 3.

### 1.2. Our Contribution

We propose the first privacy-preserving technique capable of transforming a hashed feature set (that is, a hash table and any number of weights associated with each hash), into a differentially private version of the feature set. The technique hides all the words featurized in the feature set by making them indistinguishable from all other possible words.

Differential privacy [6], [7], [8] has quickly become the state-of-the-art in privacy preservation [9], [10], [11], [12]. 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 – all hashes are equally "distant" from each other, and adding "noise" to a hash equates to 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, and some preliminary results in Section 5. Related work can be found in Section 6. We conclude in Section 7.

## 2. Background

### 2.1. Feature Hashing

Also known as "the hashing trick" [1], [2], feature hashing involves using a hash function [13] to map features to indexes. It differs from traditional hashing in that instead of using each hash as a key mapped to some value, the "trick" is that the hash itself is the value. This has two main advantages: it is computationally fast and space-efficient, and, more importantly for our purposes, *it maps a potentially infinite number of features into a known, bounded hash range*.

The hashing function is deterministic; a feature (which may just be a word in our scenario) will always be hashed to the same value $h$. Hashes cannot easily be reverse-engineered back to the original text string 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 text strings.

**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 outputted by $g(x)$ 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

Proposed in 2006 [6], differential privacy is a tractable, rigorous definition of privacy that can be quantified and reasoned about. 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." [8]

It has since been adopted as the de facto privacy standard by companies like Google [12] and Apple [14], and has been applied to numerous machine learning algorithms [9], [15]. It has also recently been adopted by the U.S. Census Bureau, who are using differential privacy for all data releases of the 2020 Census [16].

While other privacy definitions such as k-anonymity [17] and l-diversity [18] exist, they do not provide any protection from malicious users who possess auxiliary information from other sources, or offer any provable guarantees about the level of privacy being provided.

For our purposes, we can think of each "individual" or "user" as being a unique term in the text corpus $x$, and define differential privacy as follows:

**Definition 1** (Differential privacy [6])**.** *An algorithm $f(\cdot) \to M^*$ is $(\epsilon, \delta)$-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:*

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

The variable $\epsilon$ measures the maximum multiplicative change in output probabilities, and $\delta$ measures the maximum additive change (often thought of as the failure rate of the privacy guarantee [8]). 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 [19]).

Our goal is to design an algorithm $f(M) \to M^*$ that is $(\epsilon, \delta)$-differentially private.

### 2.3. Rényi Differential Privacy

Rényi differential privacy (RDP) [20] offers us another way of formulating differential privacy (DP), in terms of the Rényi divergence between two distributions. Compared to using Kullback–Leibler divergence to measure differential privacy [21], RDP better models the $\delta$ privacy risk in Definition 1, and provides us with an $\alpha$ variable that allows for a smooth trade-off between $\epsilon$ and $\delta$. Framed in terms of text corpora, RDP is defined as follows:

**Definition 2** (Rényi differential privacy [20]). *An algorithm $f(\cdot) \rightarrow M^*$ is $(\alpha, \epsilon)$-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 $D_\alpha$ is the Rényi divergence of order $\alpha > 1$ between two probability distributions $P$ and $Q$ defined over $\mathcal{R}$:*

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

It was also shown in [20] that we can convert $(\alpha, \epsilon)$-RDP into traditional $(\epsilon, \delta)$-DP in the following way:

**Lemma 1** (Converting RDP to DP [20]). *If $f(\cdot)$ obeys $(\alpha, \epsilon)$-RDP, then $f(\cdot)$ obeys $(\epsilon + \ln(1/\delta)/(\alpha - 1),\ \delta)$-DP for all $0 < \delta < 1$.*

## 2.4. User-Level Privacy

Traditionally, differential privacy 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 independently. 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 [8]. 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. 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, $|x| - |x'| = K,\ K \geq k$.

Two previous works have explored providing actual user-level differential privacy against text-based linkage attacks [22], [23]. At first glance this may sound similar to our work here, however there is key difference: these works focus on protecting the privacy of the people providing the text, rather than protecting the confidentiality of the text itself.

## 3. The Threat Model

The threat model our technique operates under is one in which a data owner has some machine learning model $M$, made up of:

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

The data owner wishes to make their model public without leaking any of the data $x$ used to train the model.

$\Theta$ can be thought of as a weight matrix, where each row (i.e., vector) index is equal to a hash $h$ in $H$, and each column is a random variable $X_i$. For clarity we describe the values in $\Theta$ 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. 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.

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 (all occurrences of the same word) in the training documents may be unknown to the attacker.

**Remark 2.** *In the scenario described, we can imagine that the attacker has gained a copy of some document from a source outside of $x$, but with one term redacted. Seeing the context surrounding the redacted word may give them some clues as to what the redacted term is. Fortunately, the definition of differential privacy already incorporates these sorts of linkage or correlation attacks, as part of the attacker's a priori knowledge. It is important to note that this scenario differs from, say, redaction, in which all non-redacted terms are directly leaked via access to $x$ (that is, no a priori knowledge is required).*

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.

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* [8], and allows our solution to be "bolted-on" [24] 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),

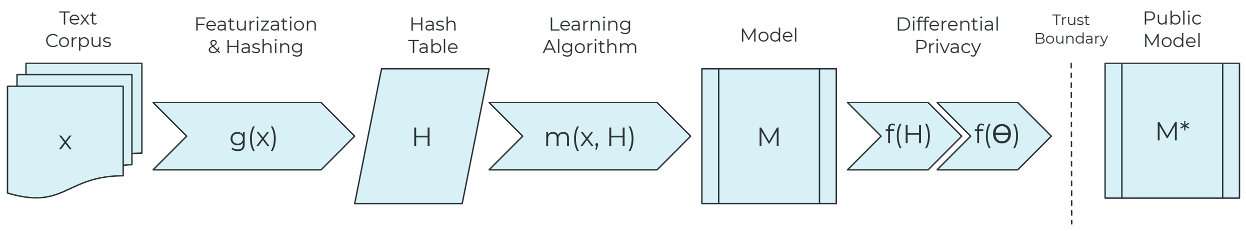

Figure 1. The process of training a model on a data set of text, then applying differential privacy on the outputted model to create a version of the model that preserves the privacy of all of the words in the text.

and Support Vector Machines (SVM's).[1] 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.

As our threat model assumes that the attacker has unlimited time and access to the whole model, this is a *non-interactive* setting [8]. 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 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 [10].

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 the features in $H$, producing some model $M$, $m(x, H) = M = \{\mathcal{L}, F\}$. The algorithm $m$ can 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 $\theta_h$, $F = \{(h, \theta_h); \forall h \in H\}$, 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 $\theta$ vectors: weight matrix $\Theta = \{\theta_h; \forall h \in H\}$, thus allowing us to write the feature set as $F = \{H, \Theta\}$.

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 [26], 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 all occurrences of one term that have been 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 list.
- 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

In order to preserve the confidentiality of each term when the terms are being mapped to hashed features, a novel differentially private solution is needed. Unlike most differential privacy techniques, there is no aggregation we can employ when it comes to the 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

1. For differentially private deep learning models, we refer the reader to [12]. 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 [25]. 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.

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 have 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, model utility remains high.

Since our goal is to protect the presence of terms, we are not concerned with an attacker observing or learning weights in the weight matrix $\Theta$, except insofar as that information could consequently reveal a term.

We use output perturbation to provide privacy: the anonymization process $f(M) \to M^*$ is performed after $M$ 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 $f(H) \to H^*$, and $f(\Theta) \to \Theta^*$. 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 (i.e., untrusted users).

See Fig. 1 for 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 $(\epsilon, \delta)$-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 $H$)**.** *Considering all possible hash values $1, \ldots, R$, 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 $H^* = \{1, 2, \ldots, R-1, R\}$ and $|H^*| = R$. The resulting hash table $H^*$ is $(\epsilon, \delta)$-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 $x$. Then, using Equation 1, we trivially have:

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

□

## 4.2. Anonymizing the Weight Matrix

Anonymizing $\Theta$ is about ensuring each $h^* \in H^*$ has a full set of plausible weights $\theta^*_h$; that is, weights that follow the distribution observed in $\Theta$, 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.

We make the hashes indistinguishable by generating synthetic weights from the same distribution as the genuine weights. The function $f(\Theta)$ first fits a $d$-dimensional mixture distribution $\mathbf{X} = \{X_1, \ldots, X_d\}$ to the realized random variates in $\Theta$. We can also write this distribution as $\mathcal{D}(\Theta)$. 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 $\theta_{h^*}$ by sampling from $\mathbf{X}$.

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 $\Theta$ are non-parametric, $f(\Theta)$ can use Kernel Density Estimation (KDE) [27] or a similar technique to perform the fit. See [28] for a recent example of using KDE for a similar purpose. 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.

## 4.3. Measuring the Privacy Cost

With $f(\Theta)$ defined, we can now measure the privacy cost of $f(\Theta)$ using Rényi differential privacy:

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

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

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

*Proof.* The calculation of $\epsilon$ and $\epsilon'$ is a direct application of Definition 2 and Lemma 1 respectively, using the concept of adjacent data sets described in Section 2.4. All vectors produced by $f(\Theta)$ are fitted to or generated by the distribution $\mathcal{D}(\Theta)$. The output of $f(\Theta)$ is always a weight matrix $\Theta^*$ of length $R$ with no missing vectors $\theta_{h^*}$, $\forall h^* \in H^*$, or missing weights $w \in \theta_{h^*}$. □

For any given adjacent weight matrix $\Theta'$ known to the attacker, the only available attack vector to detect one or more of the $K$ unknown genuine hashes is to detect $\theta$'s in $\Theta^*$ that diverge from the expected distribution $\mathcal{D}(\Theta')$, 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.

**Remark 3.** *$K$ is different for different terms. This can move adjacent distributions closer or further away, and the privacy cost $\epsilon$ will likely be higher for more frequent words, where $K$ is large. Similarly, rarer words are more protected.*

The data owner can calculate the privacy cost for specific terms if they so choose. 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 [8] applies – the costs of each term do not add up.

Moving beyond theory, the effectiveness of Claim 2 in practice depends on whether reasonable $\epsilon$ values can be achieved, and how heavily the performance of model $M^*$ is affected compared to what $M$ could achieve. We provide some preliminary results in Section 5, but first we offer some insight into why we can expect performance to remain high.

### 4.4. Model Utility

Recall from Remark 1: For any particular scenario being modelled, the output distribution $H$ will only use a subset of the universe, $H \subseteq U$, from which the training data *and* future data are both then drawn from.

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 $\theta_h$. Any weights in $\theta_h$ 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 anonymized model $M^*$'s predictions. We therefore expect model utility to remain high after making the feature set differentially private.

Note that this sampling assumption 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.

## 5. Preliminary Results

To test the viability of our solution, we had in-house lawyers annotate the definition of an "approved fund" across 373 credit, facility and loan agreements found in the EDGAR dataset [29]. While these documents are public, similar agreements in the real world could easily contain highly sensitive information.

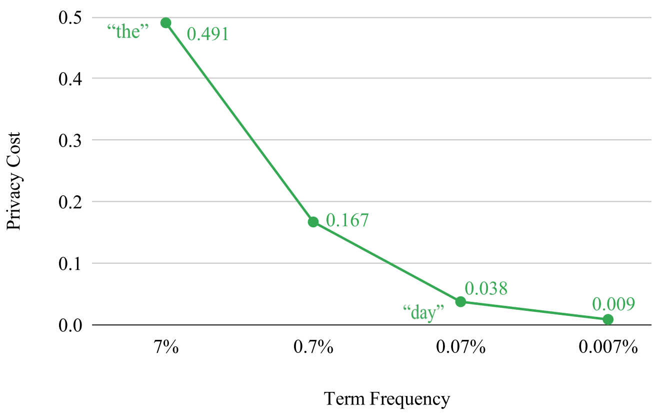

Figure 2. The privacy cost $\epsilon$ of terms at four different frequencies. The first and third frequencies are labeled "the" and "day" as examples of terms that occur at approximately that frequency.

The annotations are labeled as "relevant", and the remaining text are labeled as "not relevant". We trained a Conditional Random Field model [30] using the Passive Aggressive algorithm [31] in the CRFSuite software [32] to find the relevant text. We use MurmurHash3 [13] to perform the feature hashing, with a hash range of $R = 2^{21} \approx 2\times10^6$.

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 found in [33] 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 then summed together to form the final privacy cost $\epsilon$.

When trained on 80% of the documents and tested on the remaining 20%, the (non-private) model achieves 92% Precision and 94% Recall. After applying differential privacy and filling in the hash table, Precision drops to 89% while Recall holds steady.

As noted in Remark 3, $K$ can be calculated separately for any given term when calculating the privacy cost. Fig. 2 shows the privacy cost $\epsilon$ of terms that appear in different percentages of the total number of features (1,531,288 in this case) when $\delta = 10^{-5}$. Zipf's Law [34] 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, so $K = 107,190$ for "the". Less frequent terms appear inversely proportionally to their frequency rank, so for example the 98th most common term, "day", appears in approximately $K = 107,190/98 = 1094$ 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 $\epsilon = 0.009$.

Dwork, the creator of differential privacy, has recommended that the privacy cost remain at or below 0.1 [7], however state-of-the-art machine learning algorithms often go as high as $\epsilon = 1$ [9], $\epsilon = 2, 4, 8$ [12] or even $\epsilon = 8.6$ [11]. We find it very promising that our technique can provide guarantees as strong as $(0.167, 10^{-5})$-differential privacy for even a reasonably common term, with only minor degradation in model performance.

## 6. Related Work

**Redaction**. To the best of our knowledge, redaction is the only viable technique currently used in day-to-day operations to maintain the confidentiality of words [35], [36], [37] or documents [38]. While some work has been done on automatic redaction [37], semi-automatic redaction [35], and human-assistance tools [36], 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 [4], and seen again recently when Netflix released redacted data, but their users' privacy was still breached [5].

**Word Vector DP**. One attempt has recently been made to apply differential privacy to text representations [39] in a deep learning framework. The technique takes word vectors and converts 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. The authors 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 used in [39]. 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. Conclusion & Future Work

When models are trained on sensitive text, owners of the text want to know that none of the terms 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.

This work represents the first attempt at using differential privacy to hide which hashes are genuine 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. Our next step is to more thoroughly investigate the technique's utility and privacy cost in real-world scenarios, and extend it to handle missing weights and online learning. There are likely many more applications where our key observation – that finite output spaces can be filled in – can lead to differentially private solutions.

## References

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

[2] 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*. Montreal, Canada: ACM, 2009, pp. 1113–1120.

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

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

[5] R. Singel, "Netflix cancels recommendation contest after privacy lawsuit," p. 1, 2010. [Online]. Available: https://www.wired.com/2010/03/netflix-cancels-contest/

[6] C. Dwork, "Differential Privacy," in *Automata, Languages and Programming*, vol. 4052. Venice, Italy: Springer, 2006, pp. 1–12.

[7] ——, "Differential Privacy: A survey of results," in *Theory and Applications of Models of Computation*. Xi'an, China: Springer, 2008, pp. 1–19.

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

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

[10] B. Fung, K. Wang, R. Chen, and P. Yu, "Privacy-preserving data publishing: A survey of recent developments," *ACM Computing Surveys*, vol. 42, no. 4, pp. 1–53, 2010.

[11] 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*. IEEE, 2008, pp. 277–286.

[12] 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*. ACM, 2016, pp. 308–318.

[13] A. Appleby, "MurmurHash3," 2012. [Online]. Available: https://github.com/aappleby/smhasher

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

[15] A. D. Sarwate and K. Chaudhuri, "Signal Processing and Machine Learning with Differential Privacy," *IEEE Signal Process Magazine*, vol. 30, no. 5, pp. 86–94, 2013.

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

[17] L. Sweeney, "k-anonymity: A model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 5, pp. 557–570, 2002.

[18] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "l-diversity: Privacy beyond k-anonymity," *ACM Transactions on Knowledge Discovery from Data*, vol. 1, no. 1, p. 3, 2007.

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

[20] I. Mironov, "Rényi Differential Privacy," in *30th IEEE Computer Security Foundations Symposium*. Santa Barbara, USA: IEEE, 2017, pp. 263–275.

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

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

[23] J. Zhang, J. Sun, R. Zhang, Y. Zhang, and X. Hu, "Privacy-Preserving Social Media Data Outsourcing," in *IEEE Conference on Computer Communications*. Honolulu, USA: IEEE, 2018, pp. 1106–1114.

[24] 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)*. Chicago, USA: ACM, 2017, pp. 1307–1322.

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

[26] 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*. Genoa, Italy: European Language Resources Association, 2006, pp. 1222–1225.

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

[28] P. Burchard and A. Daoud, "Empirical Differential Privacy," *arXiv*, vol. 1910.12820, pp. 1–10, 2020.

[29] U. S. Commission and Exchange, "Electronic Data Gathering, Analysis, and Retrieval system," *SEC Docket*, vol. 118, no. 19, 2013.

[30] 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*. Morgan Kaufmann Publishers, 2001, pp. 282–289.

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

[32] N. Okazaki, "CRFsuite: a fast implementation of Conditional Random Fields (CRFs)," 2007. [Online]. Available: http://www.chokkan.org/software/crfsuite/

[33] M. Gil, F. Alajaji, and T. Linder, "Rényi divergence measures for commonly used univariate continuous distributions," *Information Sciences*, vol. 249, no. 905, pp. 124–131, 2013.

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

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

[36] 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*. Jeju Island, Korea: Springer-Verlag New York, 2015, pp. 399–410.

[37] 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*. Catalonia, Spain: Springer, 2012, pp. 173–184.

[38] 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*. Tampa, USA: IEEE, 2015, p. 6.

[39] 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*. Virtual Event, China: ACM, 2020, pp. 1813–1816.