Private Nearest Neighbors Classification in Federated Databases

Phillipp Schoppmann[1] **Adrià Gascón**[2] **Borja Balle**[3]

Abstract

Privacy-preserving data analysis in the context of federated databases distributed across multiple parties has the potential to produce richer and more accurate models than what each party can learn with their own data. Secure Multi-Party Computation (MPC) offers a robust cryptographic approach to this problem, and in fact several protocols have been proposed for various learning tasks on parametric models. In this paper we focus on k-NN, shifting the attention towards non-parametric models. We tackle several challenges arising in privacy-preserving k-NN classification on federated databases, and implement a concrete protocol for document classification. Our solution is faster than the state-of-the-art custom MPC protocol by at least one an order of magnitude.

1. Introduction

Aggregating data held by different organizations has the potential to unlock novel data analysis applications. This might be done to increase the amount of training data or its dimensionality, with the goal of producing more accurate models than the ones each organization could build using only its local dataset. However, organizations often have conflicting interests in such scenarios: while collaboration would result in more accurate predictions, disclosing their local datasets might be undesirable from a competitive point of view or directly infeasible from a legal standpoint. Such constraints rule out solutions involving an external trusted party, thus presenting a challenging scenario outside the scope of most classical approaches to privacy-preserving data analysis.

This scenario motivates the development of tools for *privacy-preserving data analysis on distributed data* which provide distributed solutions of machine learning tasks that preserve the privacy of each party's data. Problems in this space are generally parameterized by the privacy guarantees provided by the protocol. Such guarantees involve a concrete threat model describing the capabilities of an ideal adversary, and typically address the question "how much information about the dataset is revealed during the execution of the protocol?". This stems from the observation that any distributed computation requires communication between the parties, which might reveal information about the data held by each party. The goal is to protect the data contributed by each party against adversaries corrupting one or more parties involved in the computation.

Multi-Party Computation (MPC) is an area of cryptography concerned with distributed computations on private data. MPC can provide strong cryptographic privacy guarantees under several natural threat models. However, despite recent theoretical and engineering breakthroughs, using out-of-the-box MPC protocols to implement arbitrary computations on private datasets does not yet scale to real-world data analysis applications. To circumvent this limitation, it is often necessary to craft *custom* MPC protocols that go beyond what can be achieved by generic solutions. This approach has led to custom MPC protocols for linear and logistic regression training (Nikolaenko et al., 2013b; Gascón et al., 2017; Mohassel & Zhang, 2017) neural network training (Mohassel & Zhang, 2017) and evaluation (Beimel et al., 2008; Liu et al., 2017; Juvekar et al., 2018), matrix factorization (Nikolaenko et al., 2013a), PCA (Al-Rubaie et al., 2017), as well as evaluation of decision trees and naive Bayes classifiers (Bost et al., 2015).

In this paper we study the problem of k-NN classification on distributed data. Our contribution complements existing work on privacy-preserving distributed data analysis from three perspectives. First, k-NN is a non-parametric model, while all the works cited above only deal with parametric models. Despite being one of the simplest non-parametric models, k-NN enjoys remarkable theoretical properties (Chaudhuri & Dasgupta, 2014) and provides good accuracies in a wide range of applications (Efros, 2017). However, non-parametric models in general, and k-NN in particular, present several challenges from the privacy points of view. In this paper we address some of these challenges in the distributed setting. Second, our protocols provide a way to exploit the sparsity of the data. This provides obvious gains in scalability in many applications, but it is not straightforward to do in a privacy-preserving way, i.e.,

[1]Humboldt-Universität zu Berlin, Germany [2]The Alan Turing Institute, London, UK [3]Amazon Research, Cambridge, UK. Correspondence to: Phillipp Schoppmann <schoppmann@informatik.hu-berlin.de>.

without revealing the set of non-zero entries in the data held by each party. Finally, our privacy guarantees hold in a threat model where the adversary can simultaneously take over all parties but one, i.e. several organizations might try to collude to learn something about the remaining dataset. This is a distinct aspect of our protocols, as all the works cited above only protect against coalitions of at most two participants. We illustrate our techniques by implementing a standard k-NN classifier based on the cosine similarity between documents represented by their TF-IDF features. This choice is both of practical interest, and complex enough to illustrate the challenges addressed by our techniques.

Contributions. Our main contributions include novel protocols for (1) privately computing the IDF coefficients of a distributed document dataset, and (2) performing distributed multiplications on sparse private matrices. The first protocol combines MPC and differential privacy (DP) to compute and release IDFs in distributed settings in the framework of multi-party computational differential privacy (Beimel et al., 2008). The combination of MPC and DP allows our protocol to attain the same level of privacy as local DP (Kasiviswanathan et al., 2011) without incurring the degradations in accuracy associated with this notion of privacy. Releasing the IDFs of a dataset with DP enables the parties to locally compute the TF-IDF of each document in the dataset. Our second protocol allows our distributed k-NN classifier to leverage the inherent sparsity in the TF-IDF representation when computing the cosine similarity between pairs of documents or datasets of documents. Our solution relies on ideas from Private Set Intersection (Huang et al., 2012) and the recent matrix multiplication protocol by Mohassel & Zhang (2017).

We provide formal privacy proofs for all our subprotocols, as well as a utility proof of our differentially private mechanism for releasing privatized IDF values. We also empirically evaluate the scalability and accuracy of our solution to k-NN-based text classification on real-world data.

2. Distributed k-NN Classification

This section presents the different steps required to implement a distributed k-NN functionality and provides an intuitive description of our privacy requirements.

Let S be a set of servers denoted by numbers $1, \ldots, n$. We assume each server holds a private dataset Z_i with m_i training samples consisting of example-label pairs. The dataset representing the union of the datasets held by each party will be denoted by $Z = \{(x_j, y_j)\}_{j=1}^{m}$. A distributed k-NN classifier allows a client holding an unlabeled example x to interact with the servers in order to obtain a label $\hat{y}$ for x. Conceptually speaking, the classifier should emulate the behavior of a k-NN classifier where the dataset Z is held by a single party. In particular, the following steps need to be executed: (1) for $j \in \{1, \ldots, m\}$ compute the similarity score $s_j(x)$ between x and the jth example in the dataset; (2) compute the labels $\hat{y}_1, \ldots, \hat{y}_k$ corresponding to the indices $j \in \{1, \ldots, m\}$ with the top k scores; and, (3) return the majority vote $\hat{y} = \mathsf{majority}(\hat{y}_1, \ldots, \hat{y}_k)$. Note that from an algorithmic point of view, computing the similarity score between two examples requires embedding them into a feature space and computing a similarity between the corresponding feature representations.

Document Classification. At a high-level, a distributed k-NN classification protocol like the one described above must implement three functionalities: *feature extraction*, *similarity computation* and *top-k selection*. The running example we use throughout the paper is *document classification with cosine similarity between TF-IDF representations*. This problem illustrates all the challenges involved in a generic k-NN implementation and allow us to focus on scalability by tailoring our MPC protocols to a particular application. While in some cases, feature extraction can be done locally by the parties, it depends on the whole dataset for TF-IDF features. We present a protocol for extracting these privately in Section 3. Note that this step has to be performed once for the entire distributed dataset Z and is amortized over many classification queries. Similarity computation is typically the most expensive step of a k-NN classification since it scales with m and needs to be performed for every query. In Section 4 we present a protocol for scoring that can leverage sparsity in the underlying feature representation and can perform many queries in parallel. For top-k selection in a distributed setting with privacy requirements, we can use a generic MPC framework (Damgård et al., 2012), so we omit a detailed description. In our experiments, we show that this last phase contributes little to the overall running time of our classification protocol.

Privacy Requirements. Roughly speaking, to ensure privacy for the client we require that (1) only the client learns the assigned class for x, and (2) no information about the query x is learned by the servers. This must hold even when all of the servers collude. If there are multiple clients, they should similarly learn nothing about each other's documents, even when some of them collude with some or all of the servers. We formalize this requirement by employing the standard cryptographic notion of security (Goldreich, 2004; Lindell, 2016) that we repeat in Appendix A. Intuitively, this notion of privacy guarantees that the information obtained by any adversary is the same information that is leaked by an idealized setting where a trusted party performs the whole computation and returns the result to the parties.

For the servers' datasets, we require that any information revealed beyond the classification result is *differentially* *private* (Dwork et al., 2006b) with respect to every single training sample. This captures many real-world scenarios where each training sample contains data of an *individual* person and is therefore highly sensitive, while *aggregated* information about an entire population is not considered harmful. We give a formal definition of Differential Privacy in our distributed setting in Section 3.

Note that the fact that we allow collusions in our definition implies that the roles of client and server are not exclusive and settings where the client is one of the servers are covered by our model just as much as settings with just one server.

3. Private Feature Extraction

This section presents a protocol for privately computing IDF coefficients on a federated document database Z. Our protocol preserves the privacy of each document in Z by employing *multi-party computational differential privacy*. That is, the computed IDF values are differentially private, and no additional information except these IDFs is revealed by the protocol. In the context of k-NN document classification with TF-IDF features, this means each data provider and any potential client can use the privatized IDFs to compute the feature representation of any document locally.

We start by recalling the relevant definitions related to the TF-IDF feature representation and differential privacy. Then we provide our differentially private mechanism for IDF computation and discuss its implementation inside an MPC protocol. Finally we formally analyze privacy and utility.

TF-IDF Features. Recall that the TF-IDF representation of a document x is the vector $\psi(x) \in \mathbb{R}^{|\mathcal{V}|}$ with coordinates $\psi(x)(v) = \phi_{\mathrm{tf}}(v,x)\phi_{\mathrm{idf}}(v,Z)$. Here $\phi_{\mathrm{tf}}(v,x) = |x|_v$ is the *term frequency* (TF) of v in x (i.e. the number of times v occurs in x), and $\phi_{\mathrm{idf}}(v,Z)$ is the *inverse document frequency* (IDF) of v in Z (see, for example, (Manning et al., 2008)). There are several variants of the IDF coefficient; here we work with a *smoothed* version which for a word $v \in \mathcal{V}$ and a dataset Z gives $\phi_{\mathrm{idf}}(v,Z) = \log((|Z|+1)/(|Z|_v+1)) + 1$. This is in fact the default formulation used in the TF-IDF implementation provided by Scikit-learn (Pedregosa et al., 2011). Although we choose a particular definition for the sake of concreteness, our results can easily be adapted to any other IDF variant.

An important observation is that the TF part of $\psi(x)$ only depends on document x, while the IDF part depends on the whole dataset Z. This presents a challenge for computing the TF-IDF feature representation of documents belonging to a federated database. In particular, the IDF computation will require communication between the data providers, and could thus lead to privacy leaks. A naive solution would be to use an n-party MPC protocol to simultaneously compute the TF-IDF representation of all the documents in the database (see Appendix B). However, this solution presents several drawbacks, including: a high computational and communication cost; an inability to leverage the sparsity in the TF coefficients; and the fact that revealing the resulting feature representations could leak private information about documents owned by a party through their IDF part. Our alternative approach is to compute the IDF coefficients in a differentially private way and release them to the parties, who can then locally compute the TF-IDF representation of each document in their respective datasets.

Multi-Party Computational Differential Privacy. Differential privacy (DP) is a technique for privacy-preserving disclosure (Dwork et al., 2006a;b; Dwork & Roth, 2014). It prevents a potential adversary observing the output of a computation from recovering information about individual input data points, i.e., individual document in our case. This is made formal by saying that two datasets Z and Z' are neighbors if they only differ in one data point; this relation is denoted by $Z \simeq Z'$. We say that a randomized algorithm $\mathcal{A} : \mathcal{Z} \to \mathcal{W}$ is (ε, δ)-DP if for any indicator function $\chi : \mathcal{W} \to \{0,1\}$ we have

$$\forall Z \simeq Z' : \quad \mathbb{E}\big[\chi(\mathcal{A}(Z))\big] \le e^{\varepsilon}\mathbb{E}\big[\chi(\mathcal{A}(Z'))\big] + \delta \ .$$

This definition models the setting where a curator owns the input Z, executes the computation $\mathcal{A}$, and discloses the output $\mathcal{A}(Z)$.

For the purpose of providing DP in a multi-party setting one needs to modify the above definition to account for the fact that Z is distributed among several parties. Additionally, implementing DP inside an MPC protocol requires a further modification to account for the information that could be obtained by a coalition of adversarial parties involved in the computation who try to break the cryptography used in the MPC protocol. This leads to the definition of *multi-party computationally differential privacy* which has been studied by Dwork et al. (2006a); Beimel et al. (2008).

Suppose the input dataset is distributed among n parties $Z = (Z_1, \dots, Z_n)$ and write $Z \simeq_i Z'$ if $Z_i \simeq Z'_i$ and $Z_j = Z'_j$ for $i \neq j$. Suppose $\mathcal{A} : \mathcal{Z}^n \to \mathcal{W}$ is an n-party protocol and let $\mathsf{view}_{-i}(\mathcal{A}(Z))$ denote all the information observed by all parties except the ith one during the execution of $\mathcal{A}(Z)$. Then we say that $\mathcal{A}$ is (ε, δ)-MPC-DP if for all i and all $Z \simeq_i Z'$ we have

$$\mathbb{E}\big[\chi\big(\mathsf{view}_{-i}(\mathcal{A}(Z))\big)\big] \le e^{\varepsilon}\mathbb{E}\big[\chi\big(\mathsf{view}_{-i}(\mathcal{A}(Z'))\big)\big] + \delta$$

for any $\{0,1\}$-valued polynomial time algorithm χ. Note that technically speaking, this definition only makes sense for a *family* of protocols indexed by the parameter m; see (Beimel et al., 2008) for more details. The following key result states that implementing a DP algorithm inside an MPC protocol yields an MPC-DP protocol.

Theorem 1. *If $\mathcal{A}$ is $(\varepsilon, \delta_{\mathrm{DP}})$-DP with respect to $Z \simeq Z'$, then an MPC implementation of $\mathcal{A}$ where is Z distributed among n parties is $(\varepsilon, \delta_{\mathrm{DP}} + \delta_{\mathrm{MPC}})$-MPC-DP, where δ_{MPC} is a negligible function of m obtained from standard cryptographic assumptions.*

Algorithm 1: DP IDFs

```
Input: Public: $n$, $\mathcal{V}$, $c_0$, $L$, $\varepsilon_0$
Input: Private: Counts $\{|Z_i|_v\}_{v \in \mathcal{V}}$ for $i \in [n]$
Output: Privatized values $\{\tilde{c}_v\}_{v \in \mathcal{V}}$

foreach $v \in \mathcal{V}$ do
    Compute $c_v = \sum_{i=1}^{n} |Z_i|_v$
end
for $\ell = 1, \ldots, L$ do
    Sample $v \in \mathcal{V}$ with probability $\propto \exp(\varepsilon_0 c_v)$
    Sample $\eta$ from $\mathsf{Lap}(1/\varepsilon_0)$
    Release $\tilde{c}_v = c_v + \eta$
    Remove $v$ from $\mathcal{V}$
end
For each $v \in \mathcal{V}$ release $\tilde{c}_v = c_0$
```

Differentially Private IDF Computation. To compute IDFs with differential privacy we combine the exponential mechanism (McSherry & Talwar, 2007) and the Laplace mechanism (Dwork et al., 2006b). The mechanism takes as input the absolute frequencies of each word in each party's dataset Z_i. It then proceeds to aggregate these into frequencies across the whole dataset Z, yielding $c_v = |Z|_v$ for each $v \in \mathcal{V}$. The counts are used in a private top-L selection step to find L words with the largest frequencies; this is a standard construction based on the exponential mechanism (Bafna & Ullman, 2017). The mechanism then releases privatized counts $\tilde{c}_v$ for each of the selected words using the Laplace mechanism. For unselected words the mechanism outputs a default public value $\tilde{c}_v = c_0$ which is independent of the true word count. The pseudocode of our mechanism is given in Algorithm 1.

Theorem 2. *For any $\varepsilon_0 \in (0, 0.9]$ and $\delta \in [0, 1]$ the Algorithm 1 is (ε, δ)-DP with*

$$\varepsilon = \min\left\{2L\varepsilon_0, 2L\varepsilon_0^2 + \sqrt{4L\varepsilon_0^2 \log(1/\delta)}\right\} .$$

We prove Theorem 2 in Appendix C.1. By Theorem 1, we can obtain an MPC-DP protocol from Algorithm 1 by implementing it inside MPC. Computing the counts c_v only involves arithmetic operations which are simple to implement in MPC. Thus, all we need is a way for the parties to sample from the Laplace and the exponential distribution inside an MPC protocol. This requires a private distributed noise generation protocol so that parties can jointly sample from the specified distributions in a way that no coalition of $n - 1$ parties can recover η from $\tilde{c}_v$. Dwork et al. (2006a) propose a very efficient procedure to generate noise in MPC in exactly this way. Their protocol takes a small number of random bits from each party and performs operations which are efficient in MPC, such as bitwise manipulations and additions. This results in an efficient method for implementing Algorithm 1 in MPC.

Utility Analysis. A key observation about the utility of Algorithm 1 is that in a corpus of documents Z the distribution of the values of $|Z|_v$ for all $v \in \mathcal{V}$ typically follows a power-law distribution (Powers, 1998). This means there are few very frequent words and lots of infrequent words, which justifies assigning a default value c_0 to the words which are not selected by the top-L selection provided by the exponential mechanism. Our experimental evaluation (Section 5) shows that this leads to a small accuracy loss when the privatized IDFs are used for k-NN document classification.

The following result illustrates the effect of the different parameters of Algorithm 1 in the accuracy of the computed IDFs. Our analysis assumes the documents in Z are sampled i.i.d. from some unknown distribution over documents. Here we present only an informal statement of our result. A more concrete statement together with the relevant proofs are provided in Appendix C.2. The result bounds the relative error between the true vectors of IDFs ϕ_{idf} and the privatized vector $\tilde{\phi}_{\mathrm{idf}}$ computed using the counts released by Algorithm 1.

Theorem 3. *Let $c_0 = \Theta(\sqrt{m})$. If m is large enough, then with high probability we have*

$$\frac{\|\phi_{\mathrm{idf}} - \tilde{\phi}_{\mathrm{idf}}\|_1}{\|\phi_{\mathrm{idf}}\|_1} \leq \tilde{O}\left(\frac{L}{V}\frac{1}{\varepsilon_0 m} + \left(1 - \frac{L}{V}\right)\log(m)\right) .$$

Note how this result highlights an essential accuracy-privacy trade-off in the choice of L. Indeed, from Theorem 2 we see that increasing L reduces the privacy provided by the mechanism, while from Theorem 3 we see that taking a $L = V(1 - O(1/\log(m)))$ ensures the mechanism provides a constant-factor approximation to the true IDFs.

4. Private Scoring

We will now describe how to privately compute similarity scores between two feature vectors. As before, we focus on text documents with TF-IDF features, where *cosine similarity* is the measure commonly used (Manning et al., 2008). Since this essentially corresponds to a secure inner product computation between two parties, the protocol described in this section can be adapted to other similarity measures that can be reduced to inner products, such as the Euclidean distance. Our solution is especially efficient if the feature vectors are sparse, which is the case for the private TF-IDF features computed in the previous section. However, it can be generally applied to sparse vectors.

For clarity of presentation, we first focus on computing additive shares of the similarity between two sparse vectors, by essentially solving a 2-party private sparse inner product problem. Subsequently, we show how to generalize our approach to secure sparse matrix multiplication, gaining even more efficiency when computing many scores at once. At a high level, our protocol first reduces the dimensionality of the sparse input vectors to the total number of their non-zero entries, and then uses the protocol of Mohassel & Zhang (2017) to compute additive shares of their inner product.

Additive Secret Sharing. Our secure sparse multiplication protocol (Figure 1) will produce matrices $\mathbf{C_1}, \mathbf{C_2}$ that are chosen uniformly at random under the constraint that $\mathbf{C_1} + \mathbf{C_2} = \mathbf{AB}$. Since Party i obtains only $\mathbf{C_i}$, and each share individually cannot be distinguished from a random matrix, this does not leak any information about $\mathbf{C} = \mathbf{AB}$. Thus, the similarity scores computed by our protocol are not known by a single party, but secret-shared between the holders of both input document vectors. The shares are then used as inputs to a generic MPC protocol for selecting the k most similar documents, as described in Section 2. This technique is generally used to hide the values of intermediate results, and it's the basis of various MPC protocols (Beaver, 1991; Damgård et al., 2012; Mohassel & Zhang, 2017). We will refer to values shared in this way as *additively shared* and call the individual parts *additive shares*.

Index Sets. For any sparse vector $\mathbf{v}$, we write $\mathcal{I}_\mathbf{v}$ to denote the set of indexes where $\mathbf{v}$ is non-zero, and call its cardinality $l_\mathbf{v} := |\mathcal{I}_\mathbf{v}|$. We further denote the k-th element of this set (using canonical ordering) by $(\mathcal{I}_\mathbf{v})_k$.

For a matrix $\mathbf{M}$, we write $\mathsf{Col}_i(\mathbf{M})$ to denote the i-th column vector of $\mathbf{M}$, and $\mathsf{Row}_j(\mathbf{M})$ for the j-th row. In analogy to vectors, we write $\mathcal{I}_\mathbf{M}^{\mathsf{Col}} := \{i \mid \mathsf{Col}_i(\mathbf{M}) \neq \mathbf{0}\}$ for the set of set of indexes corresponding to non-zero columns of $\mathbf{M}$, likewise for rows.

Throughout this paper, we consider these sets of indexes private, as they correspond to words occurring in a text document. However, an upper bound on their cardinality (i.e., the *number* of different words) is public.

4.1. Secure Cosine Similarity

Consider two feature vectors $\mathbf{a}, \mathbf{b}$ representing text documents, where $\mathbf{a}$ is owned by Party 1 and $\mathbf{b}$ by Party 2. The cosine similarity of $\mathbf{a}$ and $\mathbf{b}$ is defined as

$$\mathsf{sim}_{\cos}(\mathbf{a}, \mathbf{b}) = \frac{\langle \mathbf{a}, \mathbf{b} \rangle}{\|\mathbf{a}\| \|\mathbf{b}\|} = \frac{\sum_{i=1}^{|\mathcal{V}|} a_i b_i}{\|\mathbf{a}\| \|\mathbf{b}\|}.$$

Note that each norm in the denominator can be computed locally by one of the parties. Additive shares of their product can then be obtained using Beaver multiplication (Beaver, 1991). Since the denominators are positive, the comparison of two scores can be done by cross-multiplication, removing the need for secure divisions. Therefore, the similarity essentially reduces to computing the inner product of two vectors owned by different parties.

Now, observe that the i-th term $a_i b_i$ of the inner product $\langle \mathbf{a}, \mathbf{b} \rangle$ is non-zero if and only if both a_i and b_i are non-zero. In fact, if the parties could make public their non-zero indexes $\mathcal{I}_\mathbf{a}$ and $\mathcal{I}_\mathbf{b}$, they could write down the values corresponding to indexes in the intersection $\mathcal{I}_\mathbf{a} \cap \mathcal{I}_\mathbf{b}$ in canonical order, resulting in vectors $\tilde{\mathbf{a}}, \tilde{\mathbf{b}}$ of shorter length $|\mathcal{I}_\mathbf{a} \cap \mathcal{I}_\mathbf{b}|$ with $\langle \tilde{\mathbf{a}}, \tilde{\mathbf{b}} \rangle = \langle \mathbf{a}, \mathbf{b} \rangle$. The inner product of $\tilde{\mathbf{a}}$ and $\tilde{\mathbf{b}}$ can then be computed using a standard MPC protocol (Mohassel & Zhang, 2017; Gascón et al., 2017).

While this would greatly increase efficiency by exploiting the sparsity of $\mathbf{a}$ and $\mathbf{b}$, it is also clear that it would leak information about both documents, namely the words they contain. To remove this leakage, we do not reveal $\mathcal{I}_\mathbf{a}$ and $\mathcal{I}_\mathbf{b}$. Instead, both parties *locally* create vectors $\hat{\mathbf{a}}, \hat{\mathbf{b}}$ containing the values at indexes in $\mathcal{I}_\mathbf{a}$ and $\mathcal{I}_\mathbf{b}$, respectively. Now, unless both documents contain the exact same words, clearly $\langle \hat{\mathbf{a}}, \hat{\mathbf{b}} \rangle \neq \langle \mathbf{a}, \mathbf{b} \rangle$. However, if we pad both vectors with zeroes to length $l_\mathbf{a} + l_\mathbf{b}$ and then permute them such that the values of indexes in $\mathcal{I}_\mathbf{a} \cap \mathcal{I}_\mathbf{b}$ somehow end up at the same position in both permuted vectors, while all others get matched to a zero, then the inner product of these permuted vectors will again be equal to $\langle \mathbf{a}, \mathbf{b} \rangle$. While a trusted third party could take both index sets and compute *correlated* permutations with the property described above, our setup does not include such a third party. Thus, we have to develop a MPC-sub-protocol that generates these permutations.

In the next section, we show how the observations made here extend to matrix multiplications. Then, we describe our two-party sparse matrix multiplication protocol in detail and state formal correctness and security theorems that we prove in Appendix D.

4.2. Computing Scores in Parallel

As we have seen, computing the cosine similarity of two documents in TF-IDF representation can essentially be reduced to a secure sparse inner product. If we want to compute the pairwise similarities of two sets of documents, this immediately reduces to secure matrix multiplication: assuming the vocabulary has size m, Party 1 arranges their l feature vectors row-wise in a matrix $\mathbf{A} \in \mathbb{Z}_q^{l \times m}$, while Party 2 uses a column-wise representation of their n documents, resulting in a matrix $\mathbf{B} \in \mathbb{Z}_q^{m \times n}$. Then, each element c_{ij} of the product $\mathbf{C} = \mathbf{AB}$ is the numerator of the similarity of Party 1's i-th document and Party 2's j-th document.

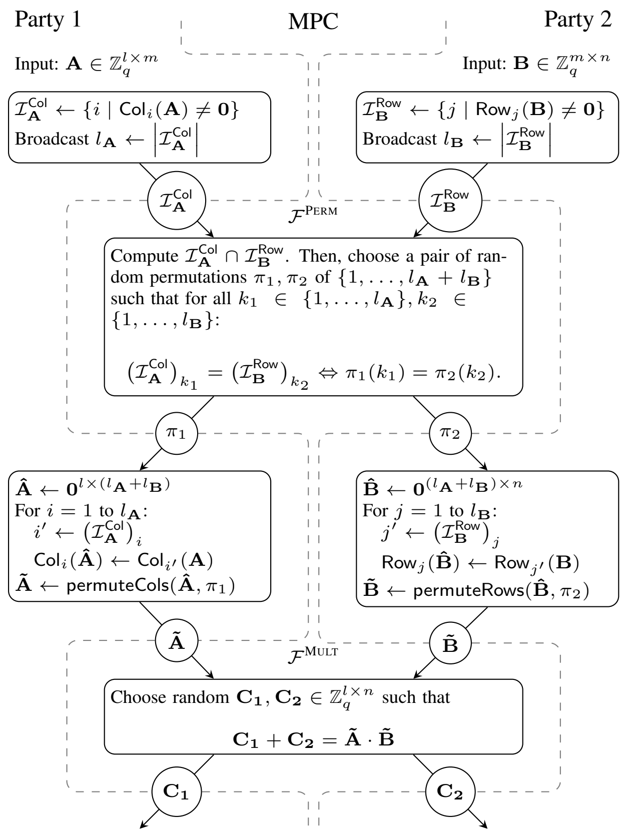
Party 1
MPC
Party 2
Input: $\mathbf{A} \in \mathbb{Z}_q^{l \times m}$
Input: $\mathbf{B} \in \mathbb{Z}_q^{m \times n}$
$\mathcal{I}_{\mathbf{A}}^{\mathsf{Col}} \leftarrow \{i \mid \mathsf{Col}_i(\mathbf{A}) \neq \mathbf{0}\}$
Broadcast $l_{\mathbf{A}} \leftarrow \left|\mathcal{I}_{\mathbf{A}}^{\mathsf{Col}}\right|$
$\mathcal{I}_{\mathbf{B}}^{\mathsf{Row}} \leftarrow \{j \mid \mathsf{Row}_j(\mathbf{B}) \neq \mathbf{0}\}$
Broadcast $l_{\mathbf{B}} \leftarrow \left|\mathcal{I}_{\mathbf{B}}^{\mathsf{Row}}\right|$
$\mathcal{I}_{\mathbf{A}}^{\mathsf{Col}}$
$\mathcal{I}_{\mathbf{B}}^{\mathsf{Row}}$
$\mathcal{F}^{\text{PERM}}$
Compute $\mathcal{I}_{\mathbf{A}}^{\mathsf{Col}} \cap \mathcal{I}_{\mathbf{B}}^{\mathsf{Row}}$. Then, choose a pair of random permutations π_1, π_2 of $\{1, \ldots, l_{\mathbf{A}} + l_{\mathbf{B}}\}$ such that for all $k_1 \in \{1, \ldots, l_{\mathbf{A}}\}, k_2 \in \{1, \ldots, l_{\mathbf{B}}\}$:
$\left(\mathcal{I}_{\mathbf{A}}^{\mathsf{Col}}\right)_{k_1} = \left(\mathcal{I}_{\mathbf{B}}^{\mathsf{Row}}\right)_{k_2} \Leftrightarrow \pi_1(k_1) = \pi_2(k_2).$
π_1
π_2
$\hat{\mathbf{A}} \leftarrow \mathbf{0}^{l \times (l_{\mathbf{A}} + l_{\mathbf{B}})}$
For $i = 1$ to $l_{\mathbf{A}}$:
$i' \leftarrow \left(\mathcal{I}_{\mathbf{A}}^{\mathsf{Col}}\right)_i$
$\mathsf{Col}_i(\hat{\mathbf{A}}) \leftarrow \mathsf{Col}_{i'}(\mathbf{A})$
$\tilde{\mathbf{A}} \leftarrow \mathsf{permuteCols}(\hat{\mathbf{A}}, \pi_1)$
$\hat{\mathbf{B}} \leftarrow \mathbf{0}^{(l_{\mathbf{A}} + l_{\mathbf{B}}) \times n}$
For $j = 1$ to $l_{\mathbf{B}}$:
$j' \leftarrow \left(\mathcal{I}_{\mathbf{B}}^{\mathsf{Row}}\right)_j$
$\mathsf{Row}_j(\hat{\mathbf{B}}) \leftarrow \mathsf{Row}_{j'}(\mathbf{B})$
$\tilde{\mathbf{B}} \leftarrow \mathsf{permuteRows}(\hat{\mathbf{B}}, \pi_2)$
$\tilde{\mathbf{A}}$
$\tilde{\mathbf{B}}$
$\mathcal{F}^{\text{MULT}}$
Choose random $\mathbf{C_1}, \mathbf{C_2} \in \mathbb{Z}_q^{l \times n}$ such that
$\mathbf{C_1} + \mathbf{C_2} = \tilde{\mathbf{A}} \cdot \tilde{\mathbf{B}}$
$\mathbf{C_1}$
$\mathbf{C_2}$

Figure 1. Secure sparse matrix multiplication. For details on the implementations of $\mathcal{F}^{\text{MULT}}$and $\mathcal{F}^{\text{PERM}}$, see Appendices B.3 and D.1, respectively.

The protocol described in Section 4.1 can be adapted to this kind of matrix inputs. To compute matrices $\tilde{\mathbf{A}}, \tilde{\mathbf{B}}$ with fewer dimensions, we have to retain all features that are non-zero in *at least one* document. To that end, Party 1 locally computes $\mathcal{I}_{\mathbf{A}}^{\mathsf{Col}} := \{i \mid \mathsf{Col}_i(\mathbf{A}) \neq \mathbf{0}\}$, and Party 2 computes $\mathcal{I}_{\mathbf{B}}^{\mathsf{Row}} := \{j \mid \mathsf{Row}_j(\mathbf{B}) \neq \mathbf{0}\}$. These index sets are then used as inputs to the functionality for generating correlated permutations, $\mathcal{F}^{\text{PERM}}$. This functionality is implemented using Yao's Garbled Circuit Protocol (Yao, 1986). It performs a PSI (Huang et al., 2012) to compute $\mathcal{I}_{\mathbf{A}}^{\mathsf{Col}} \cap \mathcal{I}_{\mathbf{B}}^{\mathsf{Row}}$ and then generates two correlated permutations π_1, π_2 that map elements of $\mathcal{I}_{\mathbf{A}}^{\mathsf{Col}} \cap \mathcal{I}_{\mathbf{B}}^{\mathsf{Row}}$ to the same indexes. $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ are then computed by first padding non-zero columns/rows with zeroes, and then applying π_1 and π_2, just like $\tilde{\mathbf{a}}$ and $\tilde{\mathbf{b}}$ in Section 4.1. The result is again obtained by using a standard MPC protocol for secure matrix multiplication with $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ as inputs. The entire sparse matrix multiplication protocol is depicted in Figure 1.

Theorem 4 (Correctness)**.** *For any* $\mathbf{A} \in \mathbb{Z}_q^{l \times m}$, $\mathbf{B} \in \mathbb{Z}_q^{m \times n}$, *let* $\tilde{\mathbf{A}}, \tilde{\mathbf{B}}$ *be constructed according to the protocol described in Figure 1. Then* $\mathbf{AB} = \tilde{\mathbf{A}}\tilde{\mathbf{B}}$.

We prove Theorem 4 in Appendix D.2. To prove security of our sparse matrix multiplication protocol, we require that the sub-protocols for $\mathcal{F}^{\text{PERM}}$ and $\mathcal{F}^{\text{MULT}}$ are secure. For the former, we use an existing protocol for Private Set Intersection (Huang et al., 2012), which we extend so that it maps indexes in $\mathcal{I}_{\mathbf{A}}^{\mathsf{Col}} \cap \mathcal{I}_{\mathbf{B}}^{\mathsf{Row}}$ to the same value, while making sure non-matching indexes get mapped to zeros. The details can be found in Appendix D.1. Our implementation is based on Yao's Garbled Circuit protocol, which was proven secure in the semi-honest adversary model (Lindell & Pinkas, 2009). For $\mathcal{F}^{\text{MULT}}$, we use the protocol of Mohassel & Zhang (2017), a simple extension of Beaver's protocol (Beaver, 1991), which is also secure in the semi-honest model. We prove the following theorem using a standard hybrid argument in Appendix D.3.

Theorem 5 (Security)**.** *Given public sparsity values* $l_{\mathbf{A}}, l_{\mathbf{B}}$ *and implementations of* $\mathcal{F}^{\text{MULT}}$ *and* $\mathcal{F}^{\text{PERM}}$ *that are secure against semi-honest adversaries, the protocol in Figure 1 implements* $\mathcal{F}^{\text{MULT}}$ *with security against semi-honest adversaries.*

5. Experiments

This section provides experimental evaluation of our text classification system. First, we report on the accuracy loss introduced by the computation of differentially private IDF values as part of the *Private Feature Extraction* described in Section 3. Next, we analyze the running time of the *Private Scoring* protocol from Section 4 and compare it against two different baselines. We also implement the top-k selection phase and measure its running time, but since it is negligible compared to the scoring protocol (less than one minute in all cases), we omit a detailed evaluation.

5.1. Accuracy

To evaluate the accuracy of our classification protocol, we used a publicly available repository of Amazon product reviews spanning May 1996 to July 2014 (He & McAuley, 2016; dat). We used the *5-core* version of the dataset containing only products with at least five reviews. From the entire dataset we extracted reviews for products in four different categories: “Clothing, Shoes and Jewelry” (clothes), “Toys and Games” (games), “Tools and Home Improvement” (diy), and “Grocery and Gourmet Food” (food). We use these product categorizations to set up a document classification problem with four classes. To construct the dataset we randomly selected 28K reviews from the four classes with a uniform class distribution. Statistics about the selected data are reported in Table 1. In order to tune the hyper-parameters of the algorithm and assess the predictive performance of the resulting models we further split the data into 70% for training, 15% for validation, and 15% for testing while maintaining the class proportions in each

class	max. $\lvert x\rvert$	avg. $\lvert x\rvert$	$\lvert\mathcal{V}\rvert$
clothes	858	56	13016
games	1725	105	20549
diy	1796	89	17995
food	1926	96	18706
all	1926	86	40558

Table 1. Statistics about the datasets used in the accuracy experiments.

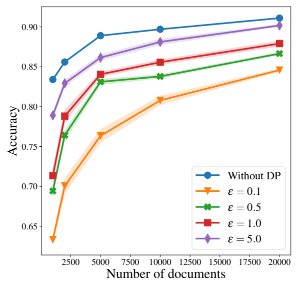

Figure 2. Classification accuracy for various privacy budgets ε.

of the subsets. When testing the effect of the amount of training data on the overall accuracy of the model we further subsample the $\sim$ 20K training examples to obtain a smaller training set.

To quantify the accuracy loss incurred by using differential privacy as a function of the privacy parameter ε_0 we report the accuracies obtained for different sizes of the training dataset, see Figure 2. Each line in this plot corresponds to the learning curve for a different setting in $\varepsilon = 2L\varepsilon_0 \in \{0.1, 0.5, 1, 5\}$, which is the privacy obtained by taking $\delta = 0$ in Theorem 2. This allows us to compare the accuracy provided by the same privacy level with different settings of L and ε_0. We also report the accuracy of the protocol which implements k-NN without differential privacy. Since differential privacy introduces randomness in the computation, we draw shades around the curves using differential privacy to report the standard deviation over 25 runs. The parameters of the protocol were tuned on the validation set, while the plots correspond to accuracies on the test set. The range of possible values considered for each parameter are as follows: number of neighbors $k \in [1, 60]$, and fraction of non-default IDFs $L/m \in \{0.01, 0.005, 0.002, 0.001\}$. These ranges were selected after an initial data exploration phase. As is common with differential privacy, we observe that for a fixed ε, the loss in accuracy with respect to the protocol that uses exact IDFs decreases as the size of the training set grows. For example, for a training dataset with $m = 20000$ documents and $\varepsilon = 1$ we observe that the accuracy loss is less than 5%.

5.2. Running Time

In this section, we evaluate our protocol in terms of online running time. This corresponds to the *Private Scoring* protocol from Section 4, as both feature extraction (which can be performed locally) and the computation of differentially private IDF values can be done in a setup phase and amortized across many classifications.

Baselines. For a significant comparison, we compare our private scoring with a state-of-the-art MPC protocol based on 2-party matrix multiplication, but without the optimizations described in Sections 3 and 4. The complete protocol is described in Appendix B.2. The basic idea is to perform a setup phase that computes additive shares of intermediate values in the scoring computation that do not depend on the client's query document. Then, the online phase mostly consists of two-party matrix multiplications and is already significantly faster than a naive approach using generic MPC. However, since additive sharing as defined in Section 4 means that the servers' arguments to these matrix multiplications are uniformly random, we cannot exploit the fact that most of the shared values are in fact zero due to the sparsity of document vectors. Therefore, we use the *dense* matrix multiplication protocol by Mohassel & Zhang (2017), which we reproduce in Appendix B.3. For a second baseline, we use the same classification protocol, but use a matrix multiplication sub-protocol based on additive Homomorphic Encryption from previous works (Murugesan et al., 2010; Bost et al., 2015). Since this protocol can exploit sparsity only of the query document, we refer to it as *semi-sparse*.

Experimental Setup. We implement our system in C++, using Obliv-C (Zahur & Evans, 2015) and SPDZ (Damgård et al., 2012) for generic Two-Party and Multi-Party Computation, respectively. For the semi-sparse protocol, we use an implementation of the additively homomorphic Damgård-Jurik encryption scheme (lib). Our timings were obtained on Azure E8s v3 instances, each having 8 vCPUs and 64 GB of RAM. For WAN experiments, we placed the instances in two different regions, East US and West Europe. All of our experiments used 3 servers and a vocabulary size of 150000, which is about the size of Aspell's `en_US-large` dictionary (asp).

Results. Figure 3 shows the results of our timing experiments. On the left, we plot the online time of the private scoring phase as a function of the number of words in the client's document. Solid lines represent the baseline protocol with dense and semi-sparse matrix multiplication, while dotted lines represent our protocol using the sparse matrix multiplication protocol from Figure 1. It can be seen that the semi-sparse protocol is slower than the dense one for all but the smallest queries. On the other hand, our sparse protocol improves upon the running time of the dense protocol by a factor of at least 10. The right plot of Figure 3 shows the online time as a function of the size of the servers' datasets for the dense and sparse protocol. It becomes apparent that the speedup introduced by our secure sparse matrix multiplication protocol is consistent across a wide range of database sizes and in both LAN and WAN settings.

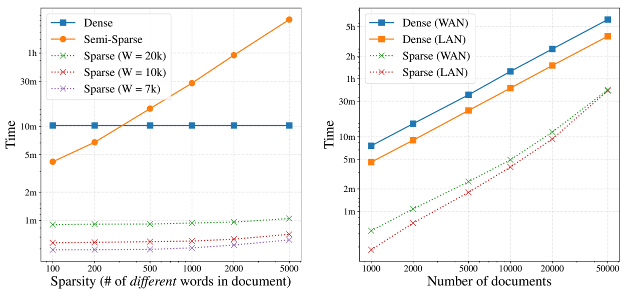

Figure 3. (Left) Running time of a single classification in a LAN vs. sparsity of the client's query document. The parameter W denotes the sparsity of each server's database, i.e., the total number of different words used in all documents of a server. Each server holds 2000 documents. (Right) Running time of a single classification as a function of the number of documents per server. The client's sparsity was fixed to 2000 and the server's sparsity was computed by extrapolating from a real-world dataset of Amazon reviews (dat).

6. Related work

As mentioned above, a remarkable difference between existing works and ours is in the threat model. More concretely, in all the contributions mentioned in the introduction, either the computation is delegated to two *non-colluding* parties – sometimes referred to as the *two-server model* in MPC – or only involves two parties.

Regarding our concrete application in private text analysis, related work can be found in the context of similar document detection (Jiang et al., 2008; Murugesan et al., 2010; Blundo et al., 2012; Buyrukbilen & Bakiras, 2013), association rule mining (Giannotti et al., 2013), document indexing and search (Bawa et al., 2009), and text summarisation (Marujo et al., 2014). All these contributions tackle the general issue of computing on a document database while preserving privacy under a variety of threat models.

Among these, the line of research relevant to our contribution is in *Secure Similar Document Detection (SSDD)* (Jiang et al., 2008; Murugesan et al., 2010; Blundo et al., 2012; Buyrukbilen & Bakiras, 2013). However, the problem considered in all of these works is limited to a two-party setting.

Finally, another trend of related work is in *privacy-preserving nearest neighbors computation* (Qi & Atallah, 2008; Songhori et al., 2015; Elmehdwi et al., 2014; Riazi et al., 2016; Li et al., 2015; Rong et al., 2016). The problem considered in the first half these contributions is in the single server setting, and hence corresponds to a two-party problem. The setting addressed by Riazi et al. (2016); Li et al. (2015); Rong et al. (2016) does allow more than two data providers, but again, their protocols are only secure in the two-server model mentioned above, where the two computing parties are assumed not to collude.

In summary, while there is previous work on parametric model training and classification, document scoring, and k-NN, the settings considered are either limited to two parties, or they require the help of semi-trusted, non-colluding external parties. In contrast, we consider the more challenging setting where privacy is guaranteed even if arbitrary collusions happen which better captures the federated databases scenarios. Note that this includes the two-server case, where all our results apply as well.

7. Conclusion

Our MPC protocol for k-NN classification achieves provable security in the federated setting with possibly colluding servers, which has not been reported in academic literature before. At the same time, our evaluation shows that it scales to real-world dataset sizes and is viable in both LAN and WAN settings. We show that by combining MPC with Differential Privacy, performance can be improved by an order of magnitude over the state-of-the-art approach, while providing a principled way to trade off between accuracy and privacy. A drawback of our current approach is the fact that the running time depends linearly on the number of training documents. Overcoming this limitation is an obvious next challenge.

Apart from classification, our private k-NN algorithm can be easily adapted to support other types of queries on distributed datasets, for example private duplicate detection, or query answering. Additionally, other document similarity measures can be implemented atop our protocol for secure two-party sparse matrix multiplication. Moreover, our protocol for sparse matrix multiplication is general in that it works on arithmetic sharings, and hence can be directly used as a building block in other applications. The latter might be of independent interest to the MPC community. In future work we plan to address some of its applications, and study further improvements using recent results on Private Set Intersection.

Beyond our concrete contributions, this work shows that hybrid solutions combining MPC and DP are a promising venue for privacy-preserving data analysis on distributed data, as carefully designed DP mechanisms for approximated functionalities can enable efficient MPC protocols.

References

Aspell dictionary creation. `http://app.aspell.net/create`.

Amazon Product Data. `http://jmcauley.ucsd.edu/data/amazon/`.

libSCAPI – The Secure Computation API. `https://github.com/cryptobiu/libscapi/`.

Al-Rubaie, Mohammad, Wu, Pei Yuan, Chang, J. Morris, and Kung, Sun-Yuan. Privacy-preserving PCA on horizontally-partitioned data. In *DSC*, pp. 280–287. IEEE, 2017.

Bafna, Mitali and Ullman, Jonathan. The price of selection in differential privacy. In *Proceedings of the 2017 Conference on Learning Theory*, Proceedings of Machine Learning Research, 2017.

Bawa, Mayank, Bayardo, Roberto J., Agrawal, Rakesh, and Vaidya, Jaideep. Privacy-preserving indexing of documents on the network. *VLDB J.*, 18(4):837–856, 2009.

Beaver, Donald. Efficient Multiparty Protocols Using Circuit Randomization. In *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pp. 420–432. Springer, 1991.

Beimel, Amos, Nissim, Kobbi, and Omri, Eran. Distributed private data analysis: Simultaneously solving how and what. In *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pp. 451–468. Springer, 2008.

Blundo, Carlo, Cristofaro, Emiliano De, and Gasti, Paolo. EsPRESSo: Efficient privacy-preserving evaluation of sample set similarity. In *DPM/SETOP*, volume 7731 of *Lecture Notes in Computer Science*, pp. 89–103. Springer, 2012.

Bost, Raphael, Popa, Raluca Ada, Tu, Stephen, and Goldwasser, Shafi. Machine learning classification over encrypted data. In *NDSS*. The Internet Society, 2015.

Buyrukbilen, Sahin and Bakiras, Spiridon. Secure similar document detection with simhash. In *Secure Data Management*, volume 8425 of *Lecture Notes in Computer Science*, pp. 61–75. Springer, 2013.

Chaudhuri, Kamalika and Dasgupta, Sanjoy. Rates of convergence for nearest neighbor classification. In *Advances in Neural Information Processing Systems*, pp. 3437–3445, 2014.

Damgård, Ivan, Pastro, Valerio, Smart, Nigel P., and Zakarias, Sarah. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pp. 643–662. Springer, 2012.

Dwork, Cynthia and Roth, Aaron. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, August 2014.

Dwork, Cynthia, Kenthapadi, Krishnaram, McSherry, Frank, Mironov, Ilya, and Naor, Moni. Our Data, Ourselves: Privacy Via Distributed Noise Generation. In *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pp. 486–503. Springer, 2006a.

Dwork, Cynthia, McSherry, Frank, Nissim, Kobbi, and Smith, Adam D. Calibrating Noise to Sensitivity in Private Data Analysis. In *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pp. 265–284. Springer, 2006b.

Dwork, Cynthia, Rothblum, Guy N, and Vadhan, Salil. Boosting and differential privacy. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, 2010.

Efros, Alexei. How to stop worrying and learn to love nearest neighbors. NIPS workshop on Nearest Neighbors for Modern Applications with Massive Data, 2017.

Elmehdwi, Yousef, Samanthula, Bharath K., and Jiang, Wei. Secure k-nearest neighbor query over encrypted data in outsourced environments. In *ICDE*, pp. 664–675. IEEE Computer Society, 2014.

Gascón, Adrià, Schoppmann, Phillipp, Balle, Borja, Raykova, Mariana, Doerner, Jack, Zahur, Samee, and Evans, David. Privacy-Preserving Distributed Linear Regression on High-Dimensional Data. *Proceedings on Privacy Enhancing Technologies*, 2017(4):345–364, October 2017.

Giannotti, Fosca, Lakshmanan, Laks V. S., Monreale, Anna, Pedreschi, Dino, and Wang, Wendy Hui. Privacy-preserving mining of association rules from outsourced transaction databases. *IEEE Systems Journal*, 7(3):385–395, 2013.

Goldreich, Oded. *The Foundations of Cryptography – Volume 2, Basic Applications*. Cambridge University Press, 2004.

He, Ruining and McAuley, Julian. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, pp. 507–517. International World Wide Web Conferences Steering Committee, 2016.

Huang, Yan, Evans, David, and Katz, Jonathan. Private set intersection: Are garbled circuits better than custom protocols? In *NDSS*. The Internet Society, 2012.

Jiang, Wei, Murugesan, Mummoorthy, Clifton, Chris, and Si, Luo. Similar document detection with limited information disclosure. In *ICDE*, pp. 735–743. IEEE Computer Society, 2008.

Juvekar, Chiraag, Vaikuntanathan, Vinod, and Chandrakasan, Anantha. Gazelle: A Low Latency Framework for Secure Neural Network Inference. *IACR Cryptology ePrint Archive*, 2018:73, 2018.

Kasiviswanathan, Shiva Prasad, Lee, Homin K, Nissim, Kobbi, Raskhodnikova, Sofya, and Smith, Adam. What can we learn privately? *SIAM Journal on Computing*, 40 (3):793–826, 2011.

Li, Frank, Shin, Richard, and Paxson, Vern. Exploring privacy preservation in outsourced k-nearest neighbors with multiple data owners. In *CCSW*, pp. 53–64. ACM, 2015.

Lindell, Yehuda. How To Simulate It – A Tutorial on the Simulation Proof Technique. *IACR Cryptology ePrint Archive*, 2016:46, 2016.

Lindell, Yehuda and Pinkas, Benny. A Proof of Security of Yao's Protocol for Two-Party Computation. *J. Cryptology*, 22(2):161–188, 2009.

Liu, Jian, Juuti, Mika, Lu, Yao, and Asokan, N. Oblivious neural network predictions via minionn transformations. In *CCS*, pp. 619–631. ACM, 2017.

Manning, Christopher, Raghavan, Prabhakar, and Schütze, Hinrich. Scoring, term weighting and the vector space model. In *Introduction to information retrieval*, pp. 100–122. 2008.

Marujo, Luís, Portelo, José, de Matos, David Martins, Neto, João Paulo, Gershman, Anatole, Carbonell, Jaime G., Trancoso, Isabel, and Raj, Bhiksha. Privacy-preserving important passage retrieval. In *PIR@SIGIR*, volume 1225 of *CEUR Workshop Proceedings*, pp. 7–12. CEUR-WS.org, 2014.

McSherry, Frank and Talwar, Kunal. Mechanism design via differential privacy. In *FOCS*, pp. 94–103. IEEE Computer Society, 2007.

Mohassel, Payman and Zhang, Yupeng. SecureML: A system for scalable privacy-preserving machine learning. In *IEEE Symposium on Security and Privacy*, pp. 19–38. IEEE Computer Society, 2017.

Murugesan, Mummoorthy, Jiang, Wei, Clifton, Chris, Si, Luo, and Vaidya, Jaideep. Efficient privacy-preserving similar document detection. *VLDB J.*, 19(4):457–475, 2010.

Nikolaenko, Valeria, Ioannidis, Stratis, Weinsberg, Udi, Joye, Marc, Taft, Nina, and Boneh, Dan. Privacy-preserving matrix factorization. In *ACM Conference on Computer and Communications Security*, pp. 801–812. ACM, 2013a.

Nikolaenko, Valeria, Weinsberg, Udi, Ioannidis, Stratis, Joye, Marc, Boneh, Dan, and Taft, Nina. Privacy-preserving ridge regression on hundreds of millions of records. In *IEEE Symposium on Security and Privacy*, pp. 334–348. IEEE Computer Society, 2013b.

Pedregosa, Fabian, Varoquaux, Gaël, Gramfort, Alexandre, Michel, Vincent, Thirion, Bertrand, Grisel, Olivier, Blondel, Mathieu, Prettenhofer, Peter, Weiss, Ron, Dubourg, Vincent, VanderPlas, Jake, Passos, Alexandre, Cournapeau, David, Brucher, Matthieu, Perrot, Matthieu, and Duchesnay, Edouard. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011.

Powers, David M. W. Applications and explanations of zipf's law. In *CoNLL*, pp. 151–160. ACL, 1998.

Qi, Yinian and Atallah, Mikhail J. Efficient privacy-preserving k-nearest neighbor search. In *ICDCS*, pp. 311–319. IEEE Computer Society, 2008.

Riazi, M. Sadegh, Chen, Beidi, Shrivastava, Anshumali, Wallach, Dan S., and Koushanfar, Farinaz. Sub-linear privacy-preserving search with untrusted server and semi-honest parties. *CoRR*, abs/1612.01835, 2016.

Rong, Hong, Wang, Huimei, Liu, Jian, and Xian, Ming. Privacy-preserving k-nearest neighbor computation in multiple cloud environments. *IEEE Access*, 4:9589–9603, 2016.

Songhori, Ebrahim M., Hussain, Siam U., Sadeghi, Ahmad-Reza, and Koushanfar, Farinaz. Compacting privacy-preserving k-nearest neighbor search using logic synthesis. In *DAC*, pp. 36:1–36:6. ACM, 2015.

Yao, Andrew Chi-Chih. How to Generate and Exchange Secrets (Extended Abstract). In *FOCS*, pp. 162–167. IEEE Computer Society, 1986.

Zahur, Samee and Evans, David. Obliv-C: A Language for Extensible Data-Oblivious Computation. *IACR Cryptology ePrint Archive*, 2015:1153, 2015.

A. Security

The following definition of security against semi-honest adversaries for two-party computation was adapted from (Goldreich, 2004). See also (Lindell, 2016). Note that in the original definition, the ensembles in (1) are also parametrized by a security parameter. For clarity of presentation, we omit it in the definition below and in our proofs.

A probabilistic two-party *functionality* is a function $\mathcal{F} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$. Let $\mathcal{F}(x,y) \overset{\text{def.}}{=} (\mathcal{F}_1(x,y), \mathcal{F}_2(x,y))$. For each party $i \in \{1,2\}$, $\mathcal{F}_i$ takes both parties' inputs x, y and returns the output $\mathcal{F}_i(x,y)$ to Party i. Let Π be a protocol that computes $\mathcal{F}$. Let $\mathsf{output}^{\Pi}(x,y) = \left(\mathsf{output}_1^{\Pi}(x,y), \mathsf{output}_2^{\Pi}(x,y)\right)$ denote the combined output of Π. Additionally, each party has a view on the protocol execution that is denoted by $\mathsf{view}_i^{\Pi}(x,y)$ and contains Party i's inputs, internal random state, and all received messages.

Now, a *simulator* $\mathcal{S}_i$ is a probabilistic polynomial-time algorithm that takes as arguments the inputs of Party i, and the functionality's output to Party i, i. e. $\mathcal{F}_i(x,y)$. Using these, $\mathcal{S}_i$ generates a view for party i. If such a simulator exists for each party and they satisfy

$$\left\{\left(\mathcal{S}_i(x, \mathcal{F}_i(x,y)), \mathcal{F}(x,y)\right)\right\}_{x,y} \overset{c}{\equiv} \left\{\left(\mathsf{view}_i^{\Pi}(x,y), \mathsf{output}^{\Pi}(x,y)\right)\right\}_{x,y} \quad (1)$$

then Π *privately computes* $\mathcal{F}$ (Goldreich, 2004). Here, $\overset{c}{\equiv}$ denotes *computational indistinguishability* as defined in (Goldreich, 2004; Lindell, 2016).

B. Baseline Protocol

B.1. Remarks on notation

Given a set of parties $S \subseteq \{1, \dots, n\}$ and a value $v \in \mathbb{Z}_q$, we write $[\![v]\!]^S$ to denote that v is additively shared among the parties in S, as defined in Section 4. We identify the shared value with the collection of shares and write $[\![v]\!]^S = ([\![v]\!]_i^S : i \in S)$ where $[\![v]\!]_i^S$ is known only to the ith party. When S is clear from the context we shall just write $[\![v]\!]$ and $[\![v]\!]_i$.

B.2. k-NN Without Exploiting Sparsity

Here, we describe a custom MPC protocol for private document classification that is already significantly faster than a naive implementation using generic MPC, but does not exploit the optimizations we describe in Sections 3 and 4. We use it as a baseline in our experiments (Section 5). The main building blocks are arithmetic share computations and an existing method for private matrix-vector multiplication.

We split the computation into two phases. The setup phase (1) is run by the servers holding the distributed dataset Z and independent of any document classifications. Hence, its cost is amortized across all executions of the classification phase (2), which is executed once for each query from the client.

The key idea behind the setup phase is to pre-compute the values

$$g_{j,v} = \phi_{\text{tf}}(v, x_j)\phi_{\text{idf}}(v, Z)^2$$
$$h_{j,v} = \phi_{\text{idf}}(v, Z)^2 \|\psi(x_j)\|^2$$

for each $x_j \in Z$ and $v \in \mathcal{V}$. These can then be used in the classification phase to simplify the computation of the scores in the k-NN classifier. It can be readily seen that for a new document d,

$$s_j(d) = \frac{\langle \psi(d), \psi(x_j) \rangle^2}{\|\psi(d)\|^2 \|\psi(x_j)\|^2} = \frac{\left(\sum_v g_{j,v}\phi_{\text{tf}}(v,d)\right)^2}{\sum_v h_{j,v}\phi_{\text{tf}}(v,d)^2} \ .$$

Thus, by arranging the pre-computed values into two $m \times |\mathcal{V}|$ matrices $\mathbf{G}$ and $\mathbf{H}$, we reduce the computation of the scores $s_j(d)$, $j \in \{1, \dots, m\}$, to the entry-wise division of the matrix-vector products $\mathbf{G}\phi_{\text{tf}}(d)$ and $\mathbf{H}\phi_{\text{tf}}(d)^2$. However, we do not compute that quocient explicitely. Using the fact that $a/b \geq c/d \implies ad \geq cd$ even allows us replace this division by more efficient multiplications.

Let us ignore the details of how to implement the setup phase – which we will explain later – and assume that at the end of the setup phase (see below), each server i holds shares $[\![\mathbf{G}]\!]_i^S$, $[\![\mathbf{H}]\!]_i^S$ of the matrices $\mathbf{G}$ and $\mathbf{H}$. We use these and a matrix vector multiplication sub-protocol MULT (Section B.3, implement the classification phase described in Protocol 2. It exploits the structure of arithmetic shares to compute shares of $\mathbf{G}\phi_{\text{tf}}(d)$ and $\mathbf{H}\phi_{\text{tf}}(d)^2$ via a series of 2-party computations. In particular, the protocol uses that if server i and the client (with index 0) collaborate to obtain arithmetic shares $[\![[\![\mathbf{G}]\!]_i\phi_{\text{tf}}(d)]\!]^{\{0,i\}}$, then using $\sum_{i=1}^n [\![\mathbf{G}]\!]_i = \mathbf{G}$ we have

$$\mathbf{G}\phi_{\text{tf}}(d) = \left(\sum_{i=1}^n [\![[\![\mathbf{G}]\!]_i\phi_{\text{tf}}(d)]\!]_0^{\{0,i\}}\right) + \sum_{i=1}^n [\![[\![\mathbf{G}]\!]_i\phi_{\text{tf}}(d)]\!]_i^{\{0,i\}},$$

where the first sum only involves shares owned by the client and each term in the second sum is owned by a single server. A similar observation holds for $\mathbf{H}\phi_{\text{tf}}(d)^2$. Now, we see that computing shares of the scores $s_j(d)$ just requires the parties to engage in a private computation to add their respective shares of $\mathbf{G}\phi_{\text{tf}}(d)$ and $\mathbf{H}\phi_{\text{tf}}(d)^2$. These correspond to shares m-dimensional vectors $\mathbf{s}$ and $\mathbf{t}$ such that

Protocol 2: Classification

```
Parties: $S' = \{0, 1, \dots, n\}$ with servers $S = \{1, \dots, n\}$ and a client 0
Input: (Server $i$) Shares $[\![\mathbf{G}]\!]_i^S$, $[\![\mathbf{H}]\!]_i^S$, and labels $Y_i$ for the documents in $Z_i$
Input: (Client) Document $d$ to classify
Output: Client obtains predicted label for $d$
  1: The client runs $\textsc{Mult}([\![\mathbf{G}]\!]_i^S, \phi_{\mathrm{tf}}(d))$ with each server $i \in S$ to produce shares $[\![[\![\mathbf{G}]\!]_i^S \phi_{\mathrm{tf}}(d)]\!]^{\{0,i\}}$
  2: Client locally computes
       $[\![\mathbf{G}\phi_{\mathrm{tf}}(d)]\!]_0^{S'} = \sum_{i=1}^{n} [\![[\![\mathbf{G}]\!]_i^S \phi_{\mathrm{tf}}(d)]\!]_0^{\{0,i\}}$
     and each server $i$ locally defines
       $[\![\mathbf{G}\phi_{\mathrm{tf}}(d)]\!]_i^{S'} = [\![[\![\mathbf{G}]\!]_i^S \phi_{\mathrm{tf}}(d)]\!]_i^{\{0,i\}}$
  3: Client runs $\textsc{Mult}([\![\mathbf{H}]\!]_i^S, \phi_{\mathrm{tf}}(d)^2)$ with each server $i \in S$ to produce shares $[\![[\![\mathbf{H}]\!]_i^S \phi_{\mathrm{tf}}(d)^2]\!]^{\{0,i\}}$
  4: Client locally computes
       $[\![\mathbf{H}\phi_{\mathrm{tf}}(d)^2]\!]_0^{S'} = \sum_{i=1}^{n} [\![[\![\mathbf{H}]\!]_i^S \phi_{\mathrm{tf}}(d)^2]\!]_0^{\{0,i\}}$
     and each server $i$ locally defines
       $[\![\mathbf{H}\phi_{\mathrm{tf}}(d)^2]\!]_i^{S'} = [\![[\![\mathbf{H}]\!]_i^S \phi_{\mathrm{tf}}(d)^2]\!]_i^{\{0,i\}}$
  5: All parties run a generic MPC protocol with inputs $[\![\mathbf{G}\phi_{\mathrm{tf}}(d)]\!]^{S'}$, $[\![\mathbf{H}\phi_{\mathrm{tf}}(d)^2]\!]^{S'}$, and $Y$ to compute the scores $s_j(d)$ and obtain the label $\tilde{y}$ for $d$, which is revealed only to the client
```

$\mathbf{s}_j/\mathbf{t}_j = s_j(d)$, i.e. the score of document $x_j \in Z$. A naive approach to find the top k scores and compute majority requires $2|Z|k$ multiplications and $k|Z|$ comparisons (recall that computing division is easily avoided). This can be easily implemented efficiently in a generic MPC protocol (cf. Section 5).

B.3. Dense Matrix Multiplication

We use the matrix multiplication protocol of Mohassel & Zhang (2017), which we show in Protocol 3 and state its correctness in Theorem 6.

The protocol works in the so-called preprocessing model, a common paradigm in MPC which delegates part of the computation to a data-independent offline phase, denoted by $\mathcal{F}^{\mathrm{OFF}}$. In Protocol 3, this refers to the computation of $\mathbf{U}$, $\mathbf{V}$ and $[\![\mathbf{UV}]\!]_i$, which can be done in advance without knowledge of $\mathbf{A}$ or $\mathbf{B}$. Such approach is applicable in all circumstances in the context of our document classification protocol, but becomes especially appropriate if the client in the classification phase is in fact one of the servers. In that case, the number of potential clients is limited and the offline phase can be performed preemptively as part of the setup phase.

Protocol 3: Dense $\textsc{Mult}$

```
Parties: 1, 2.
Input: Party 1: $\mathbf{A} \in \mathbb{Z}_q^{l\times m}$; Party 2: $\mathbf{B} \in \mathbb{Z}_q^{m\times n}$
Preprocessed: Party 1: random $\mathbf{U} \in \mathbb{Z}_q^{l\times m}$
              Party 2: random $\mathbf{V} \in \mathbb{Z}_q^{m\times n}$
              Party $i$: $[\![\mathbf{UV}]\!]_i \in \mathbb{Z}_q^{l\times n}$
Output: Party $i$ : $[\![\mathbf{AB}]\!]_i \in \mathbb{Z}_q^{l\times n}$
  1: Party 1 computes $\mathbf{U}, [\![\mathbf{UV}]\!]_1 \leftarrow \mathcal{F}^{\mathrm{OFF}}(l, m, n)$ and sends $\mathbf{E} = \mathbf{A} - \mathbf{U}$
  2: Party 2 computes $\mathbf{V}, [\![\mathbf{UV}]\!]_2 \leftarrow \mathcal{F}^{\mathrm{OFF}}(l, m, n)$ and sends $\mathbf{F} = \mathbf{B} - \mathbf{V}$
  3: Party 1 sets $[\![\mathbf{AB}]\!]_1 = \mathbf{EF} + \mathbf{UF} + [\![\mathbf{UV}]\!]_1$
  4: Party 2 sets $[\![\mathbf{AB}]\!]_2 = \mathbf{EV} + [\![\mathbf{UV}]\!]_2$;
```

Theorem 6 (Beaver, 1991; Mohassel & Zhang, 2017)**.** *Assuming a secure implementation of $\mathcal{F}^{\mathrm{OFF}}$, Protocol 3 implements $\mathcal{F}^{\mathrm{MULT}}$ with security against semi-honest adversaries.*

C. Details from Section 3

C.1. Proof of Theorem 2

Theorem 2. *For any $\varepsilon_0 \in (0, 0.9]$ and $\delta \in [0, 1]$ the Algorithm 1 is (ε, δ)-DP with*

$$\varepsilon = \min\left\{2L\varepsilon_0, 2L\varepsilon_0^2 + \sqrt{4L\varepsilon_0^2 \log(1/\delta)}\right\} .$$

Proof. Note that for any pair of neighboring datasets $Z \simeq Z'$ we have $|c_v - c'_v| \leq 1$. Thus, the analysis of the exponential mechanism implies that releasing each selected word v is ε_0-DP. Furthermore, the analysis of the Laplace mechanism implies that releasing each $\tilde{c}_v$ for each selected word is ε_0-DP. Note also that the values released for the words which are not selected are independent of the dataset Z. Thus, the result follows by applying the advanced composition theorem with $2L$ queries (Dwork et al., 2010). □

C.2. Proof of Theorem 3

Theorem 3. *Let $c_0 = \Theta(\sqrt{m})$. If m is large enough, then with high probability we have*

$$\frac{\|\phi_{\mathrm{idf}} - \tilde{\phi}_{\mathrm{idf}}\|_1}{\|\phi_{\mathrm{idf}}\|_1} \leq \tilde{O}\left(\frac{L}{V}\frac{1}{\varepsilon_0 m} + \left(1 - \frac{L}{V}\right)\log(m)\right) .$$

Recall that as input our mechanism receives counts $\{c_v : v \in \mathcal{V}\}$ estimated on a Z database with m labelled documents $x_1, \dots, x_m$ over vocabulary $\mathcal{V}$ of size $V = |\mathcal{V}|$. For the utility analysis we will assume a distribution $\mathcal{D}$ over documents such that the x_i are i.i.d. Using $\mathcal{D}$ we can define the word occurrence probabilities $p_v = \mathbb{P}_{x\sim\mathcal{D}}[v \in x]$ so that the expected counts can be written as $\mathbb{E}[|Z|_v] = \mathbb{E}[c_v] = mp_v$. Using these probabilities we can define an order on the vocabulary $\mathcal{V} = \{v_1, \dots, v_V\}$ in such a way that $p_{v_1} \geq p_{v_2} \geq \cdots \geq p_{v_V}$. To simplify our notation throughout the proof we define $\phi_v = \phi_{\text{idf}}(v, Z)$ and $\tilde{\phi}_v = \tilde{\phi}_{\text{idf}}(v, Z)$.

We start by splitting the vocabulary $\mathcal{V}$ into two parts: the set of words $\mathcal{V}_s$ selected by the exponential mechanism, and the set of not selected words $\mathcal{V}_{\bar{s}} = \mathcal{V} \setminus \mathcal{V}_s$. Note that by definition we have $|\mathcal{V}_s| = L$. The main idea behind our analysis of the error between the private and the non-private IDFs is to consider the words in $\mathcal{V}_s$ and $\mathcal{V}_{\bar{s}}$ separately. The error for words in the former depends on Laplace noise added to c_v, which the error for words in the later depends on the difference between the default value c_0 and the true count c_v. The first source of error can be controlled by bounding the noise added by the Laplace mechanism. To control the second source of error we will need to make sure that most of the words in $\mathcal{V}_{\bar{s}}$ have counts not too far from c_0.

We start by recalling well-known facts about the Laplace and the exponential mechanism.

Lemma 1. *With probability at least $1 - \gamma_l$ we have $|c_v - \tilde{c}_v| \leq \Delta_l$ simultaneously for all words $v \in \mathcal{V}_s$, where $\Delta_l = \log(L/\gamma_l)/\varepsilon_0$.*

Proof. See (Dwork & Roth, 2014, Theorem 3.8). □

Lemma 2. *Let $c_{(L)}$ denote the Lth greatest word count in $\{c_v : v \in \mathcal{V}\}$. With probability at least $1 - \gamma_e$, if $v \in \mathcal{V}_s$, then $c_v \geq c_{(L)} - \Delta_e$, where $\Delta_e = 2\log(LV/\gamma_e)/\varepsilon_0$.*

Proof. The proof follows the same structure as the classical utility analysis for the exponential mechanism (McSherry & Talwar, 2007, Lemma 7). □

Lemma 3. *With probability at least $1 - \gamma_L$ we have $c_{(L)} \geq mp_{v_L} - \Delta_L$, where $\Delta_L = \sqrt{2m\log(L/\gamma_L)}$.*

Proof. First note that by definition of $c_{(L)}$ we have

$$\begin{aligned}\mathbb{P}[c_{(L)} < mp_{v_L} - \Delta_L] &\leq \mathbb{P}\left[\min_{i\in[L]} c_{v_i} < mp_{v_L} - \Delta_L\right] \\ &\leq \sum_{i\in[L]} \mathbb{P}[c_{v_i} < mp_{v_L} - \Delta_L] \ .\end{aligned}$$

Since $\mathbb{E}[c_{v_i}] = mp_{v_i} \geq mp_{v_L}$ for any $i \in [L]$, by the Chernoff bound we have

$$\begin{aligned}\mathbb{P}[c_{v_i} < mp_{v_L} - \Delta_L] &\leq \mathbb{P}[c_{v_i} < mp_{v_i} - \Delta_L] \\ &\leq \exp\left(-\frac{\Delta_L^2}{2mp_{v_i}}\right) \\ &\leq \frac{\gamma_L}{L} \ .\end{aligned}$$

The result follows. □

Lemma 4. *Suppose we have $c_v \geq mp_{v_L} - \Delta_L - \Delta_e$ for every word $v \in \mathcal{V}_s$. Then with probability at least $1 - \gamma_l$ we have the following:*

$$\sum_{v\in\mathcal{V}_s} |\phi_v - \tilde{\phi}_v| \leq \frac{L\Delta_l}{mp_{v_L} - \Delta_L - \Delta_e - \Delta_l} \ .$$

Proof. Note that by the expression for the smoothed IDFs, for any $v \in \mathcal{V}$ we have

$$|\phi_v - \tilde{\phi}_v| = \left|\log\left(\frac{\tilde{c}_v + 1}{c_v + 1}\right)\right| \ .$$

Now, by combining the assumption $c_v \geq mp_{v_L} - \Delta_L - \Delta_e$ and Lemma 1 we see that with probability at least $1 - \gamma_l$ the following is simultaneously satisfied for all $v \in \mathcal{V}_s$:

$$\begin{aligned}\tilde{c}_v &\geq c_v - \Delta_l \\ &\geq c_{(L)} - \Delta_e - \Delta_l \\ &\geq mp_{v_L} - \Delta_L - \Delta_e - \Delta_l \ .\end{aligned}$$

Thus, using that for $0 \leq y < x$ we have $\log(x) - \log(x - y) \leq y/(x - y)$, we get

$$\begin{aligned}\left|\log\left(\frac{\tilde{c}_v + 1}{c_v + 1}\right)\right| &\leq \log\left(\frac{c_v + 1}{c_v - \Delta_l + 1}\right) \\ &\leq \frac{\Delta_l}{c_v - \Delta_l} \\ &\leq \frac{\Delta_l}{mp_{v_L} - \Delta_L - \Delta_e - \Delta_l} \ .\end{aligned}$$

The result follows from noting that $|\mathcal{V}_s| = L$. □

Lemma 5. *Suppose m is large enough to satisfy the following inequality:*

$$\begin{aligned}m \geq \Bigg(\frac{1}{p_{v_L - p_{v_{L+1}}}}\Big(&\sqrt{3p_{v_{L+1}}\log((V - L)/\gamma_{\bar{s}})} \\ &+ \sqrt{2\log(L/\gamma_L)} \\ &+ 2\log(LV/\gamma_e)/\varepsilon_0\sqrt{m}\Big)\Bigg)^2 \ .\end{aligned}$$

With probability at least $1 - \gamma_{\bar{s}}$ we have $c_{v_i} < mp_{v_L} - \Delta_L - \Delta_e$ for every $i > L$.

Proof. First note that an application of the Chernoff bound together with a union bound shows that simultaneously for all $L < i \leq V$ we have, with probability at least $1 - \gamma_{\bar{s}}$:

$$\begin{aligned} c_{v_i} &< m p_{v_i} + \Delta_{\bar{s}} \\ &\leq m p_{v_{L+1}} + \Delta_{\bar{s}} \ , \end{aligned}$$

where $\Delta_{\bar{s}} = \sqrt{3 m p_{v_{L+1}} \log((V-L)/\gamma_{\bar{s}})}$. The result follows by plugging in the definitions of Δ_L and Δ_e and observing that the constraint on m implies

$$m p_{v_{L+1}} + \Delta_{\bar{s}} \leq m p_{v_L} - \Delta_L - \Delta_e \ .$$

□

Note that the constraint on m above is satisfied by taking

$$m = \tilde{\Omega}\left(\frac{\log(LV)}{\varepsilon_0 (p_{v_L} - p_{v_{L+1}})^2}\right) \ , \qquad (2)$$

where the notation $\tilde{\Omega}(\cdot)$ hides constants and logarithmic terms in $1/\gamma_{\bar{s}}$, $1/\gamma_L$, and $1/\gamma_e$.

Lemma 6. *Suppose we have $c_v < m p_{v_L} - \Delta_L - \Delta_e$ for every word $v \in \mathcal{V}_{\bar{s}}$. Then the following holds:*

$$\sum_{v \in \mathcal{V}_{\bar{s}}} |\phi_v - \tilde{\phi}_v| \leq O\left((V-L) \max\left\{\log(c_0), \log\left(\tfrac{m p_{v_L} - \Delta_L - \Delta_e}{c_0}\right)\right\}\right) \ .$$

Proof. Recall that for every word $v \in \mathcal{V}_{\bar{s}}$ the mechanism outputs $\tilde{c}_v = c_0$. Furthermore, by assumption we have $0 \leq c_v < m p_{v_L} - \Delta_L - \Delta_e$, which yields:

$$\begin{aligned} |\phi_v - \tilde{\phi}_v| &= \left|\log\left(\frac{c_0 + 1}{c_v + 1}\right)\right| \\ &\leq \max\left\{\log(c_0 + 1), \log\left(\frac{m p_{v_L} - \Delta_L - \Delta_e + 1}{c_0 + 1}\right)\right\} \\ &= O\left(\max\left\{\log(c_0), \log\left(\frac{m p_{v_L} - \Delta_L - \Delta_e}{c_0}\right)\right\}\right) \ . \end{aligned}$$

The result follows by noting that $|\mathcal{V}_{\bar{s}}| = V - L$. □

Theorem 7. *Suppose m satisfies (2) and $c_0 = \sqrt{m p_{v_L} - \Delta_L - \Delta_e}$. Let $\gamma = \gamma_l + \gamma_e + \gamma_L + \gamma_{\bar{s}}$. With probability at least $1 - \gamma$ we have*

$$\frac{\|\phi_{\text{idf}} - \tilde{\phi}_{\text{idf}}\|_1}{\|\phi_{\text{idf}}\|_1} \leq \tilde{O}\left(\frac{L}{V}\frac{1}{\varepsilon_0 m} + \left(1 - \frac{L}{V}\right)\log(m)\right) \ .$$

Proof. By decomposing the L_1 distance according to the partition $\mathcal{V} = \mathcal{V}_s \cup \mathcal{V}_{\bar{s}}$ and plugging the bounds from Lemmas 4 and 6 we get

$$\begin{aligned} \|\phi_{\text{idf}} - \tilde{\phi}_{\text{idf}}\|_1 &= \sum_{v \in \mathcal{V}} |\phi_v - \tilde{\phi}_v| \\ &= \sum_{v \in \mathcal{V}_s} |\phi_v - \tilde{\phi}_v| + \sum_{v \in \mathcal{V}_{\bar{s}}} |\phi_v - \tilde{\phi}_v| \\ &\leq \frac{L \Delta_l}{m p_{v_L} - \Delta_L - \Delta_e - \Delta_l} \\ &\quad + (V-L) O\big(\log(m p_{v_L} - \Delta_L - \Delta_e)\big). \end{aligned}$$

Note that the conditions to apply these lemmas hold simultaneously with probability at least $1 - \gamma$ by Lemmas 1, 2, 3, and 5. Next we observe that by the definition of $\phi_{\text{idf}}(v, Z)$ we have $\|\phi_{\text{idf}}\|_1 \geq V$. The result now follows by plugging the expression for Δ_l, Δ_e, and Δ_L into the bound above, and using the notation $\tilde{O}(\cdot)$ to hide constants and logarithmic terms not involving m. □

D. Details from Section 4

The implementation of the local computations of the protocol in Figure 1 is straight-forward. As is $\mathcal{F}^{\textsc{Mult}}$, which is described by Mohassel & Zhang (2017). The remaining piece is therefore the two-party functionality $\mathcal{F}^{\textsc{Perm}}$ that generates correlated permutations.

D.1. Implementing $\mathcal{F}^{\textsc{Perm}}$

A first step is the observation that we only require the output to each party alone to be a uniformly random permutation. Thus, $\mathcal{F}^{\textsc{Perm}}$ can generate one of the permutations randomly, and derive the other deterministically from it. A detailed description of $\mathcal{F}^{\textsc{Perm}}$ is given in Algorithm 4.

Algorithm 4: $\mathcal{F}^{\textsc{Perm}}$

```
Parties: 1, 2.
Input: Party 1: $\mathcal{I}_{\mathbf{A}}^{\mathsf{Col}}$, Party 2: $\mathcal{I}_{\mathbf{B}}^{\mathsf{Row}}$
Output: Permutations $\pi_1$ (Party 1) and $\pi_2$ (Party 2)
1: Choose a permutation $\pi$ of $\{1, \ldots, l_{\mathbf{A}} + l_{\mathbf{B}}\}$ uniformly at random
2: Compute the function
     $\rho : \{1, \ldots, l_{\mathbf{A}}\} \to \{1, \ldots, l_{\mathbf{A}} + l_{\mathbf{B}}\}$,
     $\rho(i) = \begin{cases} \pi(j) & \text{if } (\mathcal{I}_{\mathbf{A}}^{\mathsf{Col}})_i = (\mathcal{I}_{\mathbf{B}}^{\mathsf{Row}})_j \\ \pi(l_{\mathbf{B}} + i) & \text{if no such } j \text{ exists} \end{cases}$
3: Extend $\rho$ to a random permutation $\pi_1$ of $\{1, \ldots, l_{\mathbf{A}} + l_{\mathbf{B}}\}$ by mapping all elements $i > l_{\mathbf{A}}$ to uniformly random unmapped elements of its codomain
4: Output $\pi_1$ to Party 1 and $\pi_2 = \pi$ to Party 2
```

By the construction of ρ in Step 2, it is clear that the condition from Figure 1, that matching indexes get mapped to the same position, holds for π_1 and π_2. We now prove that the outputs to both parties are indeed random permutations.

Theorem 8. *For any input sets $\mathcal{I}_{\mathbf{A}}^{\mathsf{Col}}$ and $\mathcal{I}_{\mathbf{B}}^{\mathsf{Row}}$ of size $l_{\mathbf{A}}$ and $l_{\mathbf{B}}$, and any party $i \in \{1,2\}$, $\pi_i = \mathcal{F}_i^{\text{PERM}}\left(\mathcal{I}_{\mathbf{A}}^{\mathsf{Col}}, \mathcal{I}_{\mathbf{B}}^{\mathsf{Row}}\right)$ is a uniformly random permutation of $\{1,\ldots,l_{\mathbf{A}}+l_{\mathbf{B}}\}$.*

Proof. **Case** $i=1$**.** By the definition in Step 2 in Algorithm 4, ρ is constructed by selecting $l_{\mathbf{A}}$ different mappings from a uniformly random permutation. Clearly, each of the $(l_{\mathbf{A}}+l_{\mathbf{B}})!\,/\,l_{\mathbf{B}}!$ possible functions is selected with equal probability. The extension of ρ in Step 3 reduces to selecting a uniformly random permutation of $l_{\mathbf{B}}$ elements. Thus, the claim follows by a simple counting argument.

Case $i=2$ follows immediately from Step 1 and $\pi_2 = \pi$.

□

We use Yao's Garbled Circuit protocol (Yao, 1986) to implement Algorithm 4. To find matching indexes in Step 2, we use the Sort-Compare-Shuffle approach of Huang et al. (2012). Since we operate in the semi-honest model, we further employ the following optimizations:

1. Instead of choosing π inside the Garbled Circuit, we let Party 2 choose it locally and use it as an input. Note that this is trivially simulatable since $\pi = \pi_2$ is Party 2's output.
2. Similarly, we reveal ρ to Party 1 and let it perform Step 3 locally. Again, simulating ρ it trivial by restricting π_1 to $\{1,\ldots,l_{\mathbf{A}}\}$.

Together with the security of Yao's protocol in the semi-honest model (Lindell & Pinkas, 2009), security of our implementation of $\mathcal{F}^{\text{PERM}}$ follows.

D.2. Correctness of Sparse Matrix Multiplication

Theorem 4 (Correctness)**.** *For any $\mathbf{A} \in \mathbb{Z}_q^{l\times m}$, $\mathbf{B} \in \mathbb{Z}_q^{m\times n}$, let $\tilde{\mathbf{A}}, \tilde{\mathbf{B}}$ be constructed according to the protocol described in Figure 1. Then $\mathbf{AB} = \tilde{\mathbf{A}}\tilde{\mathbf{B}}$.*

Proof. By construction of $\tilde{\mathbf{A}}$, for all $i \in \{1,\ldots,l\}$ and all $j \in \{1,\ldots,n\}$,

$$(\tilde{\mathbf{A}}\tilde{\mathbf{B}})_{ij} = \sum_{k=1}^{l_{\mathbf{A}}+l_{\mathbf{B}}} \tilde{a}_{ik}\tilde{b}_{kj} = \sum_{k=1}^{l_{\mathbf{A}}+l_{\mathbf{B}}} \hat{a}_{i\pi_1^{-1}(k)}\hat{b}_{\pi_2^{-1}(k)j}.$$

From the definition of $\mathcal{F}^{\text{PERM}}$, for any pair $(k_1,k_2) := \left(\pi_1^{-1}(k), \pi_2^{-1}(k)\right)$, $k \in \{1,\ldots,l_{\mathbf{A}}+l_{\mathbf{B}}\}$, one of the following cases holds.

Case 1 $k_1 \le l_{\mathbf{A}}$ and $k_2 \le l_{\mathbf{B}}$. Then there is a unique $k' \in \{1,\ldots,m\}$ such that $\left(\mathcal{I}_{\mathbf{A}}^{\mathsf{Col}}\right)_{k_1} = \left(\mathcal{I}_{\mathbf{B}}^{\mathsf{Row}}\right)_{k_2} = k'$ and $\hat{a}_{ik_1}\hat{b}_{k_2 j} = a_{ik'}b_{k'j}$.

Case 2 $k_1 > l_{\mathbf{A}}$ or $k_2 > l_{\mathbf{B}}$. Then $\hat{a}_{ik_1}$ or $\hat{b}_{k_2 j}$ are zero, and thus $\hat{a}_{ik_1}\hat{b}_{k_2 j} = 0$.

On the other hand, for any $k' \in \{1,\ldots,m\}$ with $a_{ik'}b_{k'j} \neq 0$, there is a pair (k_1,k_2) with $(\mathcal{I}_{\mathbf{A}}^{\mathsf{Col}})_{k_1} = (\mathcal{I}_{\mathbf{B}}^{\mathsf{Row}})_{k_2} = k'$ and thus a unique $k \in \{1,\ldots,l_{\mathbf{A}}+l_{\mathbf{B}}\}$ s.t. $\pi_1(k_1) = \pi_2(k_2) = k$. Therefore,

$$(\tilde{\mathbf{A}}\tilde{\mathbf{B}})_{ij} = \sum_{k' \in \mathcal{I}_{\mathbf{A}}^{\mathsf{Col}} \cap \mathcal{I}_{\mathbf{B}}^{\mathsf{Row}}} a_{ik'}b_{k'j} = (\mathbf{AB})_{ij}.$$

□

D.3. Security of Sparse Matrix Multiplication

Theorem 5 (Security)**.** *Given public sparsity values $l_{\mathbf{A}}, l_{\mathbf{B}}$ and implementations of $\mathcal{F}^{\text{MULT}}$ and $\mathcal{F}^{\text{PERM}}$ that are secure against semi-honest adversaries, the protocol in Figure 1 implements $\mathcal{F}^{\text{MULT}}$ with security against semi-honest adversaries.*

Proof. We only give the proof for the view of Party 1. By symmetry, the proof for Party 2 follows analogously. Our proof uses the standard simulation paradigm for cryptographic protocols (Goldreich, 2004; Lindell, 2016) introduced in Section A. For a functionality $\mathcal{F}$ and a protocol Π, we denote the output to player i of an execution with inputs x, y by $\mathcal{F}_i(x,y)$ and $\mathsf{output}_i^{\Pi}(x,y)$, respectively.

Using Simulators $\mathcal{S}_1^{\text{MULT}}, \mathcal{S}_1^{\text{PERM}}$, we construct a simulator $\mathcal{S}_1^{\Pi}$ in the ideal model that simulates Party 1's view on the Protocol in Figure 1, including calls to sub-protocols:

$$\begin{aligned}&\mathsf{view}_1^{\Pi}(\mathbf{A},\mathbf{B}) = \\ &\quad\left(\mathbf{A}, \mathsf{view}_1^{\text{PERM}}(\mathcal{I}_{\mathbf{A}}^{\mathsf{Col}}, \mathcal{I}_{\mathbf{B}}^{\mathsf{Row}}), \pi_1, \mathsf{view}_1^{\text{MULT}}(\tilde{\mathbf{A}}, \tilde{\mathbf{B}})\right).\end{aligned}$$

Upon receiving input $\mathbf{A}$ and output $\mathcal{F}_1^{\text{MULT}}(\mathbf{A},\mathbf{B})$, the simulator $\mathcal{S}_1^{\Pi}$:

1. computes $\mathcal{I}_{\mathbf{A}}^{\mathsf{Col}}$, the indices of non-zero columns of $\mathbf{A}$,
2. samples a permutation π_1' of $\{1,\ldots,l_{\mathbf{A}}+l_{\mathbf{B}}\}$ uniformly at random,
3. computes $\tilde{\mathbf{A}}'$ by padding and applying π_1' to $\mathbf{A}$ according to the Protocol in Figure 1, and
4. outputs $\left(\mathbf{A}, \mathcal{S}_1^{\text{PERM}}(\mathcal{I}_{\mathbf{A}}^{\mathsf{Col}}, \pi_1'), \pi_1', \mathcal{S}_1^{\text{MULT}}\left(\tilde{\mathbf{A}}', \mathcal{F}_1^{\text{MULT}}(\mathbf{A},\mathbf{B})\right)\right)$.

It remains to be shown that the output of $\mathcal{S}_1^{\Pi}$ together with the ideal functionality outputs is indistinguishable from the view on the real protocol and its outputs, i.e.,

$$\mathsf{SIM} := \big(\mathcal{S}_1^{\Pi}(\mathbf{A}, \mathcal{F}_1^{\text{MULT}}(\mathbf{A}, \mathbf{B})), \mathcal{F}^{\text{MULT}}(\mathbf{A}, \mathbf{B})\big) \overset{c}{\equiv} \big(\mathsf{view}_1^{\Pi}(\mathbf{A}, \mathbf{B}), \mathsf{output}^{\Pi}(\mathbf{A}, \mathbf{B})\big) =: \mathsf{REAL}. \quad (3)$$

We use a standard hybrid argument to show (3). That is, we construct a series of hybrid random variables $\mathsf{REAL} =: H_0, \dots, H_4 := \mathsf{SIM}$, such that $H_i \overset{c}{\equiv} H_{i-1}$ for $i \in \{1, \dots 4\}$.

H_1**:** This hybrid is identical to REAL, except that we replace the execution of $\text{MULT}(\tilde{\mathbf{A}}, \tilde{\mathbf{B}})$ with a call to $\mathcal{F}^{\text{MULT}}(\tilde{\mathbf{A}}, \tilde{\mathbf{B}})$ and use $\mathcal{S}_1^{\text{MULT}}$ to simulate the view of Party 1. The security of MULT implies that this is computationally indistinguishable from H_0.

H_2**:** Instead of $\mathcal{F}^{\text{MULT}}(\tilde{\mathbf{A}}, \tilde{\mathbf{B}})$, we call $\mathcal{F}^{\text{MULT}}(\mathbf{A}, \mathbf{B})$. Since, by definition, the components $\mathbf{C_1}, \mathbf{C_2}$ of the output of $\mathcal{F}^{\text{MULT}}(\mathbf{A}, \mathbf{B})$ are uniformly random with the constraint $\mathbf{C_1} + \mathbf{C_2} = \mathbf{AB}$, using $\tilde{\mathbf{A}}\tilde{\mathbf{B}} = \mathbf{AB}$ from Theorem 4 is enough to conclude $H_1 \overset{c}{\equiv} H_2$.

H_3**:** We use $\mathcal{F}_1^{\text{PERM}}(\mathcal{I}_{\mathbf{A}}^{\mathsf{Col}}, \mathcal{I}_{\mathbf{B}}^{\mathsf{Row}})$ to replace $\pi_1 = \mathsf{output}_1^{\text{PERM}}(\mathcal{I}_{\mathbf{A}}^{\mathsf{Col}}, \mathcal{I}_{\mathbf{B}}^{\mathsf{Row}})$ and use $\mathcal{S}_1^{\text{PERM}}$ to simulate $\mathsf{view}_1^{\text{PERM}}(\mathcal{I}_{\mathbf{A}}^{\mathsf{Col}}, \mathcal{I}_{\mathbf{B}}^{\mathsf{Row}})$. Indistinguishability from H_2 follows from the security of PERM.

H_4**:** Finally, we replace $\mathcal{F}_1^{\text{PERM}}(\mathcal{I}_{\mathbf{A}}^{\mathsf{Col}}, \mathcal{I}_{\mathbf{B}}^{\mathsf{Row}})$ with a uniformly random permutation π_1' of $\{1, \dots, l_{\mathbf{A}} + l_{\mathbf{B}}\}$, and $\tilde{\mathbf{A}}$ by $\tilde{\mathbf{A}}'$ as defined in Step 3 above. Note that this hybrid is equal to SIM.

$\mathcal{F}_1^{\text{PERM}}(\mathcal{I}_{\mathbf{A}}^{\mathsf{Col}}, \mathcal{I}_{\mathbf{B}}^{\mathsf{Row}})$ is uniformly random (Theorem 8), and thus identically distributed to π_1'. Since $\tilde{\mathbf{A}}'$ is derived from (π_1', A) in the same deterministic way that $\tilde{\mathbf{A}}$ is computed from $\big(\mathcal{F}_1^{\text{PERM}}(\mathcal{I}_{\mathbf{A}}^{\mathsf{Col}}, \mathcal{I}_{\mathbf{B}}^{\mathsf{Row}}), A\big)$, these two are also identically distributed. Therefore, $H_4 \overset{c}{\equiv} H_3$.

By transitivity of computational indistinguishability, $\mathsf{REAL} \overset{c}{\equiv} \mathsf{SIM}$ follows. □