# Privacy-aware Document Ranking with Neural Signals

Jinjin Shao, Shiyu Ji, Tao Yang
Department of Computer Science, University of California
Santa Barbara, California

## ABSTRACT

The recent work on neural ranking has achieved solid relevance improvement, by exploring similarities between documents and queries using word embeddings. It is an open problem how to leverage such an advancement for privacy-aware ranking, which is important for top $K$ document search on the cloud. Since neural ranking adds more complexity in score computation, it is difficult to prevent the server from discovering embedding-based semantic features and inferring privacy-sensitive information. This paper analyzes the critical leakages in interaction-based neural ranking and studies countermeasures to mitigate such a leakage. It proposes a privacy-aware neural ranking scheme that integrates tree ensembles with kernel value obfuscation and a soft match map based on adaptively-clustered term closures. The paper also presents an evaluation with two TREC datasets on the relevance of the proposed techniques and the tradeoffs for privacy and storage efficiency.

**ACM Reference Format:**
Jinjin Shao, Shiyu Ji, Tao Yang. 2019. Privacy-aware Document Ranking with Neural Signals. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '19), July 21–25, 2019, Paris, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3331184.3331189

## 1 INTRODUCTION AND RELATED WORK

There is a growing demand for privacy protection in Internet or cloud-based information services [14, 26]. While searchable encryption (e.g. [10, 11, 15, 34, 35]) has studied secure document matching without ranking consideration, privacy for document ranking is addressed in [1, 8, 49, 53] for linear additive scoring and in [30] for tree ensembles. On the other hand, there is a significant advancement in neural ranking methods and it is an open problem to develop privacy-aware ranking leveraging neural models.

The previous research on neural ranking falls into the following two categories: interaction-based or representation-based models [42]. The earlier work has focused on the representation-based models [24, 47] where each document and a query are separately represented as vectors through neural computation and the final ranking is based on the similarity of the two representative vectors. The recent studies have focused on interaction-based neural ranking models [16, 22, 54] where the word or term-level similarity of a query and a document is explored first based on their embedding vectors before applying additional neural computation. These studies have shown their interaction based models outperform the earlier representation-based models and thus our paper addresses privacy issues for three interaction-based models, more specifically DRMM [22], KNRM [54], and CONV-KNRM [16].

Ranking requires arithmetic calculations based on feature vectors and homomorphic encryption [20, 41] is one idea offered to secure data while letting the server perform arithmetic calculations without decrypting the underlying data. But such a scheme is still not computationally feasible when many numbers are involved, because each addition or multiplication is extremely slow, meanwhile homomorphic encryption does not support the ability of comparing two results required by ranking. Neural ranking involves more computational complexity than a linear method or tree ensembles, and hiding feature computation becomes even harder.

Recent secure neural net research [33] addresses image classification using homomorphic encryption and two-party communication with garbled circuits, to meet a different privacy requirement (clients obtain predicted results without knowing the server decision model). The online processing time can cost 3.56 seconds for classifying each image and in addition, 296MB of data needs to be communicated between a client and a server for each image. While computed scores are still un-comparable at the server side, the cost is too expensive for ranking many documents, considering each document vector as an image vector. Order-preserving encryption techniques (e.g. [3, 45]) let a server compare the encrypted results but do not support arithmetic computation on encrypted numbers. There is a line of work perturbing feature values (e.g. [28]) for classification to achieve differential privacy. Our method is aimed at document search with a goal of preserving the exact ranking model and our design uses one round of client-server communication for faster response time.

**Problem Statement.** This paper investigates how privacy consideration can be incorporated efficiently in neural ranking for top $K$ cloud data search. To our best knowledge, this paper is the first effort to address privacy-aware interaction-based neural ranking. In specific, we identify statistical document information such as word frequency and occurrence that can be leaked during neural computation. Such information is required for a number of privacy attacks studied in the previous work [9, 27, 52]. To mitigate such a leakage, our techniques replace the exact kernel value with a privacy-aware tree ensemble model [7, 25, 30, 37]. We further propose a soft match map that captures non-exact similarity signals above a threshold while providing a privacy protection using term closures and kernel value obfuscation. Our evaluation using two TREC datasets shows the relevance of the proposed tree integration can even exceed the original baselines for NDCG scores when soft match maps are not used. There is a relevance tradeoff when incorporating soft match maps.


*SIGIR '19, July 21–25, 2019, Paris, France*

ACM ISBN 978-1-4503-6172-9/19/07…$15.00
https://doi.org/10.1145/3331184.3331189

## 2 BACKGROUNDS AND PROBLEM SETTINGS

**Problem definition and feature vectors.** The problem of top $K$ document ranking is defined as follows: given a query $q$ with multiple terms and candidate documents, the server forms a feature vector for each document and ranks these documents. A ranking feature is called *raw* if it is explicitly stored (in the posting list) associated with each document, and it is called *composite* if it is computed dynamically based on raw features. Examples of raw features directly used for ranking include frequency of text words appearing in a document, and the quality score of a document. Document ranking uses both raw and composite features, which can be query-dependent and may not be precomputed in advance before query time. An example of composite features is BM25 [32] which is the summation of term frequency based raw features.

**Learning-to-rank algorithms.** Given a matched document represented by a set of raw and/or composite features, a linear ranking model uses a linear combination of document features while a tree-based ensemble produces a set of decision trees using a boosting or bagging strategy [7, 25, 37].

An interaction-based neural ranking [16, 22, 54] can be formalized as performing the following computation flow:

$$\text{RankingScore} = NN(Ker(\vec{q} \otimes \vec{d})),$$

where $\vec{q}$ and $\vec{d}$ are two sequences of embedding vectors which can be representations for a unigram or a $n$-gram in a query and a document [16, 22, 54], or can be entity embeddings for existing entities in a query and a document [55]. Those embedding vectors can be learned from one of many existing neural network models such as Word2Vec [39], GloVe [44] and relevance based word embedding [56]. Embedding vectors for $n$-grams can be generated by a convolution operation described in [16]. $NN$ is a forward neural network to compute the final ranking score.

Operator $\otimes$ is the interaction between query $q$ and document $d$ and its output is the similarity of a query term and a document term for all possible pairs from $q$ and $d$. In [16, 22, 54], cosine similarity is used for measuring term similarity with a score varying from $-1$ to 1. Let $\langle t, w\rangle$ denote the cosine similarity between the term vector of $t$ and that of another term $w$.

Operator $Ker$ represents the kernel value calculation, extracts term-level matching signals based on the similarity of all term pairs from a query and a document, and generates a vector of real values being taken as input for the forward neural computation. There are two methods for kernel computation. In the *Histogram Pooling* [22] method, there are $R$ kernels and each kernel associates with an interval within $[-1, 1]$, e.g., $[0.5, 0.6)$. The kernel value of the $j$-th kernel is the number of similarity values that fall into the $j$-th interval $[b_j, b_{j+1}]$: $K_j(t, d) = \sum_{w \in d} \mathbb{1}_{b_j \le \langle t, w\rangle < b_{j+1}}$. For kernel pooling, we follow a definition in [16, 54] that each kernel associates with a Radial Basis Function (RBF). The RBF kernel for the $j$-th kernel is defined by $\mu_j$ and $\sigma_j$. Symbols $\mu_j$ and $\sigma_j$ denote the mean and standard deviation respectively. Let exp be the natural exponential function and log be the natural logarithm function. The kernel value of the $j$-th kernel for a query term $t$ is:

$$K_j(t, d) = \sum_{w \in d} \exp(-\frac{(\langle t, w\rangle - \mu_j)^2}{2\sigma_j^2}).$$

The output of kernel computation is a kernel vector of size $R$:

$$(\sum_{t \in q} \log K_1(t, d), \cdots, \sum_{t \in q} \log K_R(t, d))^T.$$

**Privacy requirement and threat model**. A client owns all data and wants to outsource the search service to a cloud server which is honest-but-curious, i.e., the server will honestly follow the client's protocol, but will also try to learn any private information from the client data. The client builds an encrypted but searchable index and lets a server host such index. This paper does not consider the dynamic addition of new documents to the existing index, assuming the client can periodically overwrite the index in a cloud host server to include new content. To conduct a search query, the client sends several encrypted keywords and related information to the server. Our design only uses one round of client-server communication since multi-round active communication between the server and client (e.g. [23, 40]) incurs a much higher communication cost and response latency.

The biggest threat is the leakage of query and document plaintext. A server can also be interested in query access pattern and statistical information even if the query terms are encrypted. Finally the result patterns such as the overlapping of document IDs in multiple queries may also be interesting. This paper is focused on providing a privacy protection to avoid the leakage of document plaintext and also important feature values during ranking process.

The previous works on plaintext or query attacks in [9, 27, 52] assume the adversary knows partial information on term occurrence in addition to a subset of plaintext documents. By preventing the leakage of the term occurrence information of documents in a hosted dataset, threats from these attacks can be removed or greatly alleviated. Islam et al. [27] proposed the query recovery attack called IKK which can be revised to launch a plaintext attack to identify some words in an encrypted document collection. This assumes that the adversary is the server who has some prior knowledge as follows. 1) The server knows plaintext of $n_a$ words that appear in this document collection, but does not know the encrypted word IDs. 2) The server knows co-occurrence probabilities of these $n_a$ words in this document collection. This can be approximated by using a public dataset. 3) The server has obtained encrypted IDs of documents that contain a subset of known $n_a$ words. With the above three pieces of information, the server is able to recover the encrypted word IDs for a good percentage of these $n_a$ words, and detect the set of document IDs containing these encrypted word IDs. Cash et al. [9] has improved the plain text recoverability of the IKK attack with extra information such as term frequency, and pointed out that the inverted index or occurrence probabilities for a set of words can be inferred when knowing the term frequency of English words in each document. There are also attacks exploiting leaked document similarities [52]. Their attacks only work if the adversary knows occurrence frequency and co-occurrence frequency of selected terms in the entire document set.

Since all of the above attacks require term occurrence and use term frequency if possible, this paper will analyze the information leakage of interaction-based neural ranking methods on term frequency and occurrence in documents, and extend or redesign some of their components with a goal of hiding such statistical text information. It is worthy to note that some advanced techniques, e.g.

ranking based on Convolutional Neural Network [43], require term positions and such information can be leaked during computation. Then term occurrences in a document can be easily inferred by a server. Thus this paper does not investigate such ranking models.

## 3 LEAKAGE ANALYSIS AND DESIGN CONSIDERATIONS

We first examine the possible leakage of information in interaction-based neural ranking in terms of term occurrence and frequency.

**Hiding term vectors.** As mentioned above, there are three steps in the interaction-based model: interaction between query and document terms, kernel value calculation, and forward neural network computation. Our first thought is to hide term vectors using a hashing function while preserving cosine similarities (or other similarity metrics) between vectors. Such a protection can mask identities of term vectors; however, if a server is allowed to observe the result of interaction between a query term $t$ and each term $w$ of document $d$, it can easily infer frequencies and occurrences of all query terms as follows: $TF(t,d) = \sum_{w \in d} \mathbb{1}_{\langle t,w \rangle = 1}$, where $\mathbb{1}_{\langle t,w \rangle = 1}$ equals to 1 if $\langle t, w \rangle = 1$ and 0 otherwise.

Given the fact that we cannot store masked term vectors or explicitly store the interaction matrix elements, we resort the following strategy where the result of interaction between query and document terms is not computed by the server.

**Kernel-level protection.** Our next design idea is to provide a kernel-level protection of privacy by precomputing kernel values in advance. Thus only the output of $Ker$ is exposed to the server, which hides the vector computation process and the result of interaction. Namely the owner of the dataset (as the client in our case) precomputes this kernel vector first for each term $t$ that may interact with a document $d$, $\vec{f}_{t,d} = (a_1, , \cdots, a_j, \cdots, a_R)^T$ where $a_j = \log K_j(t,d)$. These vectors, after precomputed by a client, are uploaded and stored in a server. During the run time with a query $q$, then the kernel vector for this query can be constructed as $\sum_{t \in q} \vec{f}_{t,d}$. Then such a kernel vector is injected as an input to the forward neural computation step.

**Leakage from kernel vectors.** Unfortunately as we analyze below, the above kernel-level protection can still leak term frequency, which yields the leakage of term occurrence.

Notice for both histogram pooling and kernel pooling, the last kernel $K_R(t,d)$ is a special kernel representing the exact match of a query term with document terms. Without loss of generality, let this special one be the $R$-th kernel in this paper. For histogram pooling, an interval defined as $[1,1]$ is associated with this special kernel. This means that last element $a_R$ in $\vec{f}_{t,d}$ gets updated by one whenever a query term $t$ exactly matches a document term in $d$. As a result, the server can infer the term frequency of a query term in any document by observing the result of $a_R$.

Proposition 3.1. *Given query term $t$ and document $d$, in histogram pooling for the $R$-th kernel whose interval is* $[1,1]$*, term frequency* $TF(t,d) = \sum_{w \in d} \mathbb{1}_{\langle t,w \rangle = 1} = \exp(a_R)$.

For the $R$-th kernel derived with kernel pooling [16, 54], $\mu_R$ is 1.0, and $\sigma_R$ is chosen to be a small positive real number, e.g., 0.001. We define the maximum cosine similarity between any two different terms in a vocabulary $V$ where $V$ is the collection of all terms in the given dataset:

$$\bar{S} = \max_{t,w \in V, t \neq w} \langle t, w \rangle .$$

The following theorem shows a server can still approximate term frequency $TF(t,d)$ using the value of last kernel $a_R$.

Theorem 3.2. *Given a query term $t$ and a document $d$, in kernel pooling for the $R$-th kernel, if* $\bar{S} < 1.0 - \sqrt{2\sigma_R^2 \ln \frac{n}{\epsilon}}$*, then* $|TF(t,d) - \exp(a_R)| < \epsilon$*, where $\epsilon$ is a small real value.*

In our tested datasets in Section 6, $\bar{S}$ is below 0.9. Using the above theorem, condition $\bar{S} \leq 0.947 < 1.0 - \sqrt{2\sigma_R^2 \ln \frac{n}{\epsilon}}$ is true when $\sigma_R \leq 0.01$, $\epsilon = 0.01$, and $n \leq 10,000$, and a server can easily infer the frequency of a term in a document. Thus we are unable to use $a_R$ and the next section will present a solution to address this.

It should be noted that the above analysis is true for computing the interaction between a unigram query term with all unigrams of a document in DRMM [22] and KNRM [54]. When computing the interaction of a $h$-gram from a query and a $g$-gram from a document where $h \neq g$ in CONV-KNRM [16], the cosine similarity of such a pair cannot be 1. Thus for CONV-KNRM, we only need to worry about the interaction of two terms with the same gram length.

## 4 PRIVACY-AWARE NEURAL RANKING

In this section, we propose three techniques for privacy-aware neural ranking: 1) replace the exact match kernel value with a traditional ranking method that uses exact word matching; 2) provide a soft match index (we call it soft match map) to include a set of kernel values with similar terms while enhancing privacy; 3) obfuscate kernel values in the soft match map.

### 4.1 Replacement of the Exact Match Kernel

Neural signals can be considered to be composed of two parts: exact match component represented by last kernel value $K_R(t,d)$ and the soft match component represented by $K_1(t,d), \cdots, K_{R-1}(t,d)$ as shown in Fig. 1(a). Since Theorem 3.2 indicates that the root cause of term frequency leakage is the inclusion of kernel value $K_R$, we propose to drop this kernel value and compensate it by the including of a traditional ranking method that has better privacy protection, as illustrated in Figure 1(b).

We adopt a privacy-aware learning-to-rank tree ensemble model in [30] that encodes raw features with comparison preserving mapping (CPM) and derives a tree ensemble using encoded raw features. Raw ranking features mainly based on exact term matching are not leaked to the server. During our evaluation, these exact text matching features include BM25 for query words that appear in the title, BM25 for query words that appear in the body, and the proximity features [2, 19, 51, 57] with the minimum, maximum, and average of the squared min distance reciprocal of query word pairs in the title or in the body. We use word-pair or n-gram based features as a substitute to avoid composite proximity features based on word positions through arithmetic calculation. For ClueWeb09, extra raw features include PageRank and a binary flag indicating whether a document is from Wikipedia.

The above replacement is applied for computing the unigram-to-unigram interaction in DRMM and KNRM. It is also applicable

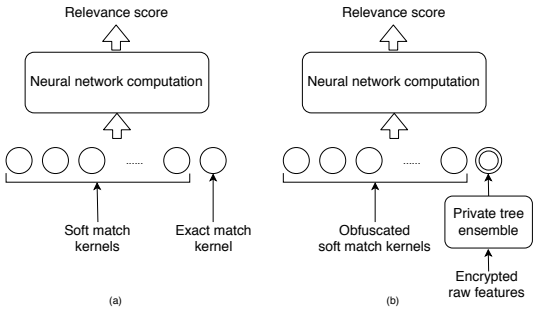


**Figure 1: Replacement of the exact match kernel**

to the interaction of a $h$-gram query term with a $h$-gram document term for CONV-KNRM. We discuss more on this in Section 6. This replacement comes with two advantages. First, it removes the source of term frequency leakage in $a_R$ since an adversary is not able to recover feature values encoded with CPM [30]. Second, it can potentially boost ranking performance. Currently there is no known method to combine a traditional ranking method with a neural ranking model for a better relevance. A tree ensemble method has been proven to be effective before neural models gain more attention. For example, in the Yahoo! learning-to-rank challenge [13] in 2010, all winners have used some forms of tree ensembles. This replacement can provide a natural way to effectively combine a tree ensemble with a representation based neural model and we will evaluate the relevance impact in Section 6.

## 4.2 Obfuscation of Kernel Values

Even though we have precomputed the kernel value computation to avoid the leakage of term frequency to the server, there still exists a term frequency attack described in Appendix A when the histogram pooling is used. In that attack, the frequency of a term queried can be uncovered by a server if it is able to find all or many of encrypted keys $(t, d)$ where $t$ is a soft or exact term of document $d$ in a soft match map. While we have not found a term frequency attack for the kernel pooling, we still want to be cautious.

To minimize the chance of leaking exact kernel values, we propose a many-to-one mapping to obfuscate kernel vector values. Intuitively, if kernel vector values from multiple different term document interactions are indistinguishable, the adversary has to make random guesses on real kernel vector values. In specific, we add the ceiling function to convert the floating point number to an integer in forming the kernel vector, and this change allows the revised soft match kernel values to accomplish $k$-anonymization [18, 50]. For the $j$-th element of a kernel vector based on $R$ kernels,

$$a_j = \begin{cases} \lceil \log_r(K_j(t,d)) \rceil, & \text{if } K_j(t,d) \geq 1, \\ 1, & \text{otherwise,} \end{cases}$$

where the logarithmic base $r$ is a privacy parameter that can be adjusted. As we show later in Section 5.2, the anonymous factor $k$ is $r^{R-1} - 1$, since all zero kernel values are converted to 1. A large $r$ value will add more anonymity for privacy while it may degrade the relevance performance due to the lack of value differentiation in kernel vectors.

In DRMM, at the forward neural computation stage, the kernel values are re-scaled by document frequency weights of query terms. We use the same many-to-one function discussed above to obfuscate these weights. Our evaluation shows there is no visible relevance difference when $r = 10$.

## 4.3 Soft Match Maps

Kernel vector values are precomputed before query processing and such offline processing can be done efficiently on a parallel platform and/or with LSH approximation [31]. However, it is too expensive to store kernel vectors for all possible $(t, d)$ pairs. For example, suppose there are $R = 20$ kernel values per term-document pair, for a dataset with 2M documents and a vocabulary of 250K terms, all these kernel vectors require 20 terabytes of space if each kernel value is stored using a reduced precision with 2 bytes. Although data compression may optimize this cost, the optimized storage cost is still excessively high.

Inspired by the inverted index, we propose a soft match index structure that contains kernel vectors of term-document pair $(t, d)$ only if term $t$ is reasonably related to document $d$, above a similarity threshold. This index data structure, called *soft match map*, has a key-value representation. The key of each entry is a hashed term-document pair $(t, d)$, and the value is kernel vector $\vec{f}_{t,d}$ generated from $Ker$ for a term-document pair. Notice $a_R$ for $R$-th kernel is removed as discussed in Section 4.1 for interaction between a query term and a document term under the same gram length. We call a term for a document this map as *exact* term if this term appears in this document. If this term is not in this document but it is included in the map due to its similarity to another term in this document, we call this term as *soft* term of this document. We will access how a soft match map is accessed during search in Section 5.

Even though terms and documents in a soft match are encrypted and identified through numeral IDs, an adversary may infer the the occurrence of terms in a document which is a critical piece of information for privacy attack discussed in Section 2. In order to minimize the chance of leaking term occurrence in a document, we introduce the notion of *$\tau$-similar term closure*.

Definition 1. *A set of terms $C$ under vocabulary $V$ is called a $\tau$-similar term closure if for any term $t \in C$ and there exists another term $w \in V$ such that $\langle t, w \rangle \geq \tau$, then $w \in C$.*

Here $V$ is the collection of terms in a given dataset and we will discuss a clustering algorithm shortly that groups a set of terms as a term closure.

Definition 2. *A soft match map SMM is closed under term closures if for any $(t, d) \in SMM$, for any $w$ in the same term closure of $t$, $(w, d) \in SMM$.*

There are two advantages of a closed soft match map. 1) From the relevance point view, a document that contains a word which is similar to a query word that can get some rank credit as the privacy-aware ranking only uses this soft match map to identify semantically related documents. 2) As shown in the next section, an adversary would have a hard time to detect if a word ID that appears in the document or not because other similar words in its term closure have all appeared in the soft match map. We will analyze its privacy implication based on the notion of statistical indistinguishability in the next section.

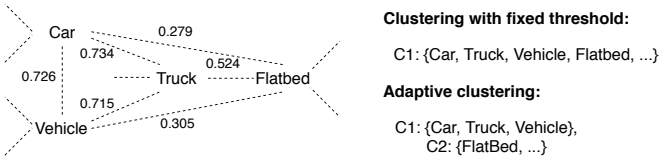


**Figure 2: Two clustering methods for term closure**

In the rest of the paper, we will assume a soft match map is closed. We now describe how to partition a term vocabulary of a dataset into a disjoint set of term closures. There are tradeoff factors to consider in controlling the size of term closures. With a larger size, the soft match map will accommodate more similar terms that can improve relevance, while creating more challenges for an adversary to distinguish and detect which term IDs in a soft match map appear in a document. On the other hand, a larger size demands more storage to host a soft match map. In the following, we discuss two algorithms that derive a disjoint set of term closures.

**Clustering with a fixed similarity threshold.** Given a clustering threshold $\tau$, we cluster all terms in a closure $C$ using a transitive closure computation as follows: if $t \in C$, and $\langle t, w \rangle \geq \tau$, then $w \in C$. This approach uses a uniform threshold for all clusters. With a small clustering threshold value, some clusters can have a very big size, which can result in a very large storage demand. With a large threshold value, some of term closures have a very small size, not big enough for the privacy purpose.

**Adaptive clustering with multiple thresholds and closure size control.** Given $p$ as a targeted closure size, and $m$ sorted clustering thresholds $\tau_1 > \tau_2 > \cdots > \tau_m$. We first apply a similarity clustering with a fixed threshold $\tau_1$. We remove all clusters with the size no less than $p$. For the remaining terms, we apply a similarity clustering with a fixed threshold $\tau_2$. Repeat this process until all cluster sizes are no less than $p$ or we have applied clustering with all thresholds. This adaptive clustering provides a flexibility to group a sufficient number of similar terms in each closure while yielding a reduced storage demand. In our evaluation, $p = 5$ is used.

Figure 2 shows the difference of the above two clustering for a partial similarity graph of 4 terms from one of our testing datasets. The edges represent pairwise similarity scores. With fixed similarity threshold at 0.5, all four terms "Car", "Truck", "Vehicle", and "Flatbed" are clustered transitively into the same term closure. With adaptive clustering using a threshold set {0.9, 0.8, 0.7, 0.6, 0.5}, and closure target size 3, term "Flatbed" is not grouped with "Car", "Truck", and "Vehicle". This is because that when threshold 0.7 is used, terms "Car", "Truck", and "Vehicle" are grouped, reaching closure size 3 and they are removed to form a separate closure. "Flatbed" will then be grouped with other terms in the rest of the graph (which is not shown in this figure).

# 5 LEAKAGE AND PRIVACY OF SOFT MATCH MAPS

## 5.1 Search Process and Leakage Profile

The top $K$ search process is described as follows. A client first sends randomized tokens for query terms and related information (called trapdoor information [1]) to a server, and these terms include query unigrams and/or multi-grams when needed. The server cannot map tokens into query terms since tokens are randomized. After the server receives tokens for all the query terms, a privacy-aware document retrieval model based on [1, 5, 6, 10, 11] produces a set of document candidates for further ranking. The server first computes the keys to access the CPM-encoded features and run a tree ensemble, where the leakage profile is studied in [1]. Then the server computes the keys to access the soft match map, and fetches associated kernel vectors to drive the forward neural computation.

Depending on the query processing semantics and privacy requirement, there are two methods to pass the query term list and document list to neural ranking, and each of which has a different leakage profile. The first method is based on the private search work of [1, 11]. Based on trapdoor information sent from a client, the server can compute and obtain a key $(t, d)$ to access the soft match map. Essentially the server obtains a list of keys representing $(t, d)$ pairs where $t$ is a query term ID and $d$ is a candidate document ID, but the server cannot decompose each key into two parts to obtain its term ID. Finally the server returns a ranked list of encrypted document IDs and these IDs are available in the map values of keys. For this case, we list the leakage profile as follows.

- Initially the server does not know any key $(t, d)$ for the map. After processing queries, the server gradually learns more keys of the soft match map.
- Once the server knows a key $(t, d)$, it can retrieve the soft kernel vector of this key and an encrypted document ID.

The second method is based on the work in [5, 6, 10], the server receives a list of query term IDs and a list of candidate documents ID. It then computes each key $(t, d)$ and learns about term ID $t$ and document ID $d$. For this case, there is additional leakage.

- Gradually after processing more queries, the server learns keys and kernel vectors, and also is able to build a partial soft inverted index for all queried terms. Based on the soft inverted index, the server is able to estimate partial document similarities based on the overlapping degree of soft terms between two documents.
- Once all terms and documents are queried or processed, the server is able to build a complete soft forward index and soft inverted index. Namely give a list of documents that contain a soft term, and give a list of soft terms included in a document. By using the soft term postings, the server learns the membership information of each soft term closure.

Since the information listed above is slowly leaked as more queries are processed, we propose to re-index the dataset periodically and replace the index in the cloud with different term IDs during each update to mitigate the chance of letting the server exploit the entire soft inverted index. The next two subsections study the privacy properties with respect to exact term occurrence even if the entire soft index is leaked to a server adversary, since such information is required for the plaintext attacks discussed in the last part of Section 2. We will also discuss k-anonymity for kernel vector values.

## 5.2 $k$-anonymization of Kernel Value Vectors

We show our obfuscation mapping achieves $k$-anonymization (a standard notion from privacy literature [18, 50]) with respect to the kernel value $K_j(t, d)$.

Definition 3. *[50] Let $V$ be a vector of real values representing $R-1$ kernel values namely, $V = [K_1(t, d), K_2(t, d), \cdots, K_{R-1}(t, d)]$.*

*Let $V' = \mathcal{F}(V)$ where $\mathcal{F}$ is a transform function. $V'$ is k-anonymous if and only if, for any $V'$, there are at least k different $V$ such that $V' = \mathcal{F}(V)$. An algorithm $\mathcal{F}$ is called k-anonymization algorithm, if it outputs a k-anonymous vector $V'$ for any soft matching signals vector $V$.*

It is easy to show that the application of the above ceiling function to the above logarithmic mapping is $k$-anonymous. In the first group, there are $r$ values $0, 1, 2, ..., (r-1)$ mapped to 1, and in the $k$-th group, there are $r^k - r^{k-1}$ values $r^{k-1}, ..., r^k - 1$ mapped to $k$ for $k > 1$. Since $r \geq 2$, $k$-th group has $r^k - r^{k-1} > r(r-1) \geq r$ values mapped to $k$, there are at least $r$ values being mapped into the same value for each group. As there are $(R-1)$ kernels, given a sequence of $\lceil \log_r(K_j(t,d)) \rceil$, there are $r^{R-1}$ different sequences of $K_j(t,d)$. Thus here we can confirm that there are $r^{R-1}-1$ sequences of kernel values that the adversary cannot distinguish from the real one. In our evaluation, we choose $r = 10$ and $R = 20$. Thus $r^{R-1} = 10^{19}$, which is a very large number.

Proposition 5.1. *The logarithmic mapping with ceiling obfuscation is k-anonymous without respect to soft match kernel values, where $k = r^{R-1}$.*

### 5.3 Obfuscation of Exact Term Occurrence

How strong a closed soft match map can be in avoiding the leakage of term occurrence? If an adversary including a server tries to detect an exact term of a document from a soft match map which contains both soft and exact terms with encrypted term IDs, we argue that other similar soft terms from the same closure behavior closely by looking at the structure and the kernel values in this soft match map for this document and the adversary should have a hard time to differentiate. We justify this argument based on the notion of statistical indistinguishability used in the cryptographic literature [4, 21].

Definition 4. ***$\varepsilon$-statistical indistinguishability.*** *Any two distributions $P$ and $Q$ over a finite set $U$ are $\varepsilon$-statistically indistinguishable if their statistical distance $SD(P,Q) \leq \varepsilon$, where statistical distance is defined as [4, 21] $SD(P,Q) = \frac{1}{2}\sum_{x \in U} |P(x) - Q(x)|$.*

If two distributions $P$ and $Q$ are $\varepsilon$-statistically indistinguishable, then no adversary can successfully distinguish the samples from $P$ and $Q$ with probability more than $\frac{1}{2} + \varepsilon$ (Theorem 3.11 in [4]). For our context, we define the distribution of a soft kernel value vector as a discrete distribution over the soft kernels with respect to a given term paired with a different document. In particular, given the kernel vectors for documents $d$ and $d'$: $\vec{f}_{t,d} = (a_1, , \cdots, a_j, \cdots, a_R)^T$, $\vec{f}_{t,d'} = (a'_1, , \cdots, a'_j, \cdots, a'_R)^T$, where each $a_i$ and $a'_i$ may be obfuscated, the statistical distance between $\vec{f}_{t,d}$ and $\vec{f}_{t,d'}$ is defined as $SD(\vec{f}_{t,d}, \vec{f}_{t,d'}) = \frac{1}{2}\sum_{i=1}^{R-1} |a_i - a'_i|$.

Definition 5. ***$\varepsilon$-statistically indistinguishable soft match map.*** *A soft match map is $\varepsilon$-statistically indistinguishable if for any document $d$ and for any term $w$ in $d$, for any document $d'$ constructed by replacing each term $w$ in $d$ with a subset of the closure $C$ such that $w \in C$, the soft kernel values of $(w,d)$ and $(w,d')$ are $\varepsilon$-statistically indistinguishable.*

In ClueWeb09 Dataset, using our algorithm where logarithmic base for kernel value obfuscation is $r = 10$, the derived soft match map is a $\varepsilon$-statistically indistinguishable with $\varepsilon$ being 0.004.

Theorem 5.2. *Given a $\varepsilon$-statistically indistinguishable soft match map for document set $D$ in which each term closure has at least $p$ terms, if a server can derive the document occurrence of exact terms with $N$ term-document pairs, there exist $(2^p - 1)^N$ different document sets $\tilde{D}$ such that the keys (term-document pairs) of its soft match map for $D$ and $\tilde{D}$ are identical, while soft kernel values of these two maps for corresponding terms are $\varepsilon$-statistically indistinguishable.*

Theorem 5.2 shows that for any soft match map generated by a document set $D$, there are at least $(2^p - 1)^N$ document sets $\tilde{D}$ with different term occurrences and co-occurrences, while having very similar soft match maps. In [9], the adversary queries 150 single-word queries to launch an IKK attack. Assuming the average posting length is 100, the number of term-document pairs guessed for a document set $D$ is $N$=15,000. If $p = 2$, then there are at least $3^{15000} \approx 10^{7157}$ document sets $\tilde{D}$ to correspond to the guessed inverted index. The adversary has to choose the correct one from these $10^{7157}$ options. Note that any two such options disagree on term occurrence for at least one document. Hence there are $10^{7157}$ term occurrence profiles for the adversary to choose from, which is very unlikely to succeed.

## 6 EVALUATION

Here we evaluate relevance scores of the proposed privacy-aware neural ranking techniques using two TREC datasets and assess tradeoffs of privacy and relevance, and storage and time cost.

**Datasets, features, and training.** We use the following TREC test collections to do evaluations. 1) Robust04 uses TREC Disks 4 & 5 (excluding Congressional Records), which has about 0.5M news articles. 250 topic queries are collected from TREC Robust track 2004. 2) ClueWeb09-Cat-B uses ClueWeb09 Category B with 50M web pages. There are 150 topic queries from the TREC Web Tracks 2009, 2010 and 2011. Spam filtering is applied on ClueWeb09 Category B using Waterloo spam score with threshold 60. During indexing and retrieval, Krovetz Stemming [36] is used for both queries and documents.

Candidate documents with their encrypted feature vectors are retrieved from the inverted index built for the above datasets, following the work in [1, 12, 34]. For a privacy-aware tree ensemble, we use CPM [30] with LambdaMART based on RankLib 2.5 [17]. In each fold of training ranking model, 5-fold cross validation is used to select the best model based on NDCG@20, varying the number of leaves from 2 to 30 and the number of trees from 100 to 500.

To evaluate the impact of feature choices with the integration of the tree ensemble on the final ranking relevance, we have three options of features listed as follows.

1) **G0 with term frequency features:** BM25 scores for query terms in the title field, and BM25 scores for query terms in the body field of each document. TF-IDF scores for query terms in the title field, and TF-IDF scores for query terms in the body field of each document. 2) **G1 with term frequency and proximity features:** All features from G0, the squared minimum distance reciprocal of query term pairs in the title field, and the squared minimum distance reciprocal [2, 19, 51, 57] of query term pairs in the body

field of each document. 3) **G2 with term frequency, proximity, and page quality features:** All features from G1, PageRank, and a binary flag indicating whether a document is from Wikipedia. This group is only for ClueWeb09 Category B.

The baseline models are DRMM, KNRM, and CONV-KNRM trained with 5-fold cross validation. We also choose a variant of CONV-KNRM, denoted by CONV-KNRM*. For CONV-KNRM*, we only use the interactions between query unigrams and document unigrams, between query unigrams and document bigrams, and between query bigrams and document unigrams. The interaction between query bigrams and document bigrams are not included to reduce storage space need. For both histogram pooling and kernel pooling, R=30 kernels are used. All soft match kernels are equally distributed in the cosine range. In kernel pooling, $\sigma$ is 0.10 for all soft match kernels. In CONV-KNRM, n-gram length is 2, and the number of CNN filters is 128 as used in the original work. All word embedding vectors are pre-trained, and are fixed in KNRM, CONV-KNRM and CONV-KNRM*. We use 300 dimension word embedding vectors trained on TREC Disks 4 & 5 or ClueWeb09 Category-Cat-B with Skip-gram + Negative sampling model [39]. All terms that appear less than 5 times are removed from embedding training.

We assess the use of the following 3 techniques denoted with T, O, and C where T stands for the replacement of the exact match kernel with LambdaMART/CPM, O stands for kernel value obfuscation, and C stands for using a closed soft match map. Notation $A/T$ means ranking $A$ with technique T while $A/TOC$ means ranking $A$ with all 3 techniques. All NDCG [29] values are within confidence interval ±0.01 with p-value $< 0.05$.

**Impact of replacing the exact match kernel with a LambdaMART/CPM tree ensemble.** Table 1 shows the NDCG relevance of the three neural ranking models as the baseline and relevance after the replacement of the exact match kernel with LambdaMART/CPM based on the three groups of features G0, G1, and G2. Soft match maps and kernel value obfuscation are not incorporated. The boldfaced numbers are the highest NDCG scores within each ranking model. From this table we observe that neural ranking with the use of LambdaMART/CPM trees outperforms the original baseline in NDCG at all Positions 1, 3, 5, and 10 for ClueWeb. For example, compared with CONV-KNRM, CONV-KNRM/T with G2 can improve NDCG@1, NDCG@3, NDCG@5 and NDCG@10 by 2.23%, 2.45%, 2.67% and 4.23% on ClueWeb. For Robust04, tree ensemble integration delivers up to 3.91% improvement for CONV-KNRM and up to 8.02% for KNRM, but degrades by up to -8.18% for DRMM. Comparing the use of $G0$, $G1$, and $G2$ for neural ranking integration, $G2$ is still most effective for ClueWeb and $G1$ is most effective for Robust04, which shows traditional signals still make a good contribution.

We examine NDCG scores reported in the previous work. For ClueWeb09-Cat-B, NDCG scores at Positions 1, 10 and 20 are 0.294, 0.289, 0.287 with CONV-KNRM in [16]. Our numbers are slightly higher, which can be caused by different data processing. Notice NDCG@20 in our run is 0.2950.

By comparing CONV-KNRM and CONV-KNRM*, the absence of bigram-bigram interaction does yield a loss of NDCG score. For example, the loss is 8.56%, 6.29%, 7.04% and 6.96% for ClueWeb at Positions 1, 3, 5, and 10, respectively. That represents a tradeoff of privacy and relevancy. Adding the tree ensemble integration, most NDCG scores for CONV-KNRM*/T can be on par with those of CONV-KNRM and are 4.31% better for ClueWeb09 at Position 10.

**Impact of kernel value obfuscation.** Table 2 shows the impact of kernel value obfuscation on NDCG scores incorporating LambdaMART/CPM with G2 for ClueWeb and G1 for Robust04. We choose two different logarithmic base $r$ here: 5 and 10. Overall, the relevance with $r = 5$ is slightly better than $r = 10$, while both of them result in degradation in ranking accuracy compared with no obfuscation. For NDCG@1, the degradation with $r = 10$ is 1.7% for CONV-KNRM, 6.36% for KNRM, and 6.3% for DRMM. For CONV-KNRM*, its degradation is relatively smaller.

**Trade-offs between relevancy and storage efficiency.** Table 3 studies the impact of using a closed soft match map with two clustering methods for term closures under different thresholds. This table is for CONV-KNRM*/TOC only as this model delivers the highest NDCG scores with all privacy preserving techniques. For example, with the obfuscation base being 10, and the clustering threshold being 0.7, for DRMM/TOC in ClueWeb, NDCG@5 and NDCG@10 are 0.2704 and 0.2720, respectively. For KNRM/TOC in ClueWeb, NDCG@5 and NDCG@10 are 0.2972 and 0.2912, respectively. In Table 3, Column 6 in the middle shows the storage need in gigabytes to store a soft match map and related data after clustering with fixed thresholds while last column on the right is the storage demand with adaptive clustering. Each entry has two numbers X(Y). Y is the total storage for unigram-unigram interaction while X is the total storage for unigram-unigram, unigram-bigram, and bigram-unigram interaction. Number Y also represents the amount of storage space needed for KNRM and DRMM.

While clustering with a fixed threshold yields some relevance improvement over adaptive clustering, it requires an excessive amount of storage space. The adaptive clustering threshold 0.7 is a good trade-off with an acceptable storage space for hosting the ClueWeb dataset. In this setting, the relevance of CONV-KNRM* is on par with the original CONV-KNRM baseline, lower than CONV-KNRM/T. That represents a tradeoff of relevancy, privacy and storage cost. It still requires 7.627TB space and we can use a number of high-end SSDs with parallel I/O. The latest high-end SSD products from Intel [48] and Samsung [38, 46] have achieved 10-15$\mu s$ IO latency with up-to 750K I/O operations per second. Thus for a soft match map hosted at a high-end SSD, the I/O access time of processing one query can still be reasonable.

**Estimation of online query processing time.** The online query processing time cost consists of 3 phases: 1) private result retrieval and preliminary ranking, 2) private tree ensemble scoring , 3) neural ranking. Based on [1], Phase I costs 460 ms on average for ClueWeb. For re-ranking top 1,000 candidate documents, there are about up-to 10K IO operations needed to fetch kernel vectors and features, and the total I/O time for accessing SSDs can take around 100 to 150ms with the above fast SSD performance parameters. Our experiments show that private tree ensemble scoring takes less than 2ms and all three neural models with TOC take about 10 ms or less. Thus overall the online query processing time is about 572ms to 622ms on average for ClueWeb. Notice that CONV-KNRM requires computation of term vectors and interaction matrices which could take 4-5 seconds. Thus even though our design pays extra cost in space, it does remove the expensive time spent for interaction computation.

**Table 1: Relevance impact of replacing exact match kernel with a tree ensemble**

| Model | Feature group for ensemble | ClueWeb09-Cat-B NDCG@1 | NDCG@3 | NDCG@5 | NDCG@10 | Robust04 NDCG@1 | NDCG@3 | NDCG@5 | NDCG@10 |
|---|---|---|---|---|---|---|---|---|---|
| LambdaMART/CPM | G0 | 0.2498 | 0.2702 | 0.2571 | 0.2415 | 0.4819 | 0.4465 | 0.4257 | 0.3982 |
| | G1 | 0.2818 | 0.2725 | 0.2688 | 0.2653 | **0.5181** | **0.4610** | **0.4346** | **0.4044** |
| | G2 | **0.2893** | **0.2828** | **0.2873** | **0.2827** | - | - | - | - |
| DRMM | Baseline | 0.2586 | 0.2659 | 0.2659 | 0.2634 | 0.5049 | **0.4872** | **0.4747** | **0.4528** |
| DRMM/T | G0 | 0.2635 | 0.2721 | 0.2623 | 0.2503 | 0.4993 | 0.4594 | 0.4425 | 0.4134 |
| | G1 | 0.2838 | 0.2778 | 0.2772 | 0.2645 | **0.5114** | 0.4658 | 0.4501 | 0.4158 |
| | G2 | **0.2887** | **0.2857** | **0.2822** | **0.2793** | - | - | - | - |
| KNRM | Baseline | 0.2663 | 0.2739 | 0.2693 | 0.2681 | 0.4983 | 0.4812 | 0.4647 | 0.4527 |
| KNRM/T | G0 | 0.2736 | 0.2804 | 0.2798 | 0.2725 | 0.5158 | 0.4908 | 0.4768 | 0.4592 |
| | G1 | **0.3036** | 0.2974 | 0.2951 | 0.2903 | **0.5382** | **0.5063** | **0.4906** | **0.4673** |
| | G2 | 0.2999 | **0.3097** | **0.3154** | **0.3147** | - | - | - | - |
| CONV-KNRM | Baseline | 0.3155 | 0.3124 | 0.3126 | 0.3085 | 0.5373 | 0.4875 | 0.4742 | 0.4586 |
| CONV-KNRM/T | G0 | 0.3031 | 0.3088 | 0.3154 | 0.3052 | 0.5402 | **0.5057** | 0.4894 | 0.4643 |
| | G1 | **0.3254** | 0.3187 | 0.3177 | 0.3099 | **0.5556** | 0.5042 | **0.4927** | **0.4693** |
| | G2 | 0.3225 | **0.3200** | **0.3210** | **0.3216** | - | - | - | - |
| CONV-KNRM* | - | 0.2884 | 0.2927 | 0.2906 | 0.2870 | 0.5007 | 0.4702 | 0.4601 | 0.4510 |
| CONV-KNRM*/T | G0 | 0.3038 | 0.2998 | 0.2962 | 0.2933 | 0.5149 | 0.4827 | 0.4768 | 0.4535 |
| | G1 | **0.3276** | 0.3099 | 0.3099 | 0.3117 | **0.5404** | **0.5006** | **0.4892** | **0.4657** |
| | G2 | 0.3175 | **0.3122** | **0.3239** | **0.3218** | - | - | - | - |

**Table 2: Impact of kernel value obfuscation with different logarithmic bases**

| Model | Obfuscation base (r) | ClueWeb09-Cat-B NDCG@1 | NDCG@3 | NDCG@5 | NDCG@10 | Robust04 NDCG@1 | NDCG@3 | NDCG@5 | NDCG@10 |
|---|---|---|---|---|---|---|---|---|---|
| DRMM/TO | Yes(10) | 0.2703 | 0.2731 | 0.2740 | 0.2732 | 0.5078 | 0.4681 | 0.4449 | 0.4157 |
| | Yes(5) | 0.2769 | 0.2757 | 0.2753 | 0.2722 | 0.5110 | 0.4669 | 0.4446 | 0.4221 |
| | No | 0.2887 | 0.2857 | 0.2822 | 0.2793 | 0.5114 | 0.4658 | 0.4501 | 0.4158 |
| KNRM/TO | Yes(10) | 0.2808 | 0.2929 | 0.2947 | 0.2906 | 0.5117 | 0.4639 | 0.4393 | 0.4130 |
| | Yes(5) | 0.2875 | 0.2968 | 0.2988 | 0.2971 | 0.5100 | 0.4686 | 0.4451 | 0.4164 |
| | No | 0.2999 | 0.3097 | 0.3154 | 0.3147 | 0.5382 | 0.5063 | 0.4906 | 0.4673 |
| CONV-KNRM*/TO | Yes(10) | 0.3121 | 0.3097 | 0.3165 | 0.3100 | 0.5221 | 0.4980 | 0.4906 | 0.4623 |
| | Yes(5) | 0.3178 | 0.3067 | 0.3161 | 0.3100 | 0.5306 | 0.4987 | 0.4893 | 0.4613 |
| | No | 0.3175 | 0.3122 | 0.3239 | 0.3218 | 0.5404 | 0.5006 | 0.4892 | 0.4657 |

**Table 3: NDCG score and storage demand for CONV-KNRM*/TOC**

| Similarity threshold | Clustering with fixed threshold NDCG@1 | NDCG@3 | NDCG@5 | NDCG@10 | Storage (GB) | Adaptive clustering NDCG@1 | NDCG@3 | NDCG@5 | NDCG@10 | Storage (GB) |
|---|---|---|---|---|---|---|---|---|---|---|
| Robust04 | | | | | | | | | | |
| 0.3 | 0.5225 | 0.4974 | 0.4915 | 0.4621 | 45,021 (2,144) | 0.5127 | 0.4892 | 0.4845 | 0.4582 | 1,512 (72) |
| 0.5 | 0.5154 | 0.4883 | 0.4780 | 0.4543 | 24,793 (1,181) | 0.5078 | 0.4845 | 0.4756 | 0.4498 | 1,133 (54) |
| 0.7 | 0.4886 | 0.4644 | 0.4486 | 0.4169 | 287 (13) | 0.4899 | 0.4608 | 0.4414 | 0.4110 | 278 (13) |
| 0.9 | 0.4953 | 0.4594 | 0.4415 | 0.4091 | 261 (12) | 0.4913 | 0.4558 | 0.4397 | 0.4090 | 261 (12) |
| ClueWeb09-Cat-B | | | | | | | | | | |
| 0.3 | 0.3136 | 0.3078 | 0.3149 | 0.3091 | $1.7 \cdot 10^6$ (82,308) | 0.3052 | 0.3069 | 0.3142 | 0.3120 | 46,811 (2,231) |
| 0.5 | 0.3073 | 0.3069 | 0.3114 | 0.3069 | $1.3 \cdot 10^6$ (61,877) | 0.3056 | 0.3037 | 0.3113 | 0.3103 | 35,742 (1,705) |
| 0.7 | 0.3064 | 0.3048 | 0.3122 | 0.3104 | 16,568 (792) | 0.3067 | 0.3012 | 0.3088 | 0.3060 | 7,627 (366) |
| 0.9 | 0.3069 | 0.3041 | 0.3105 | 0.3074 | 7,369 (354) | 0.2963 | 0.3025 | 0.3117 | 0.3083 | 7,369 (354) |

## 7 CONCLUSION

The main contribution of this paper is a privacy-aware neural ranking scheme integrated with a tree ensemble for server-side top $K$ document search. The key techniques include the replacement of the exact kernel with a tree ensemble, a soft match map using obfuscated kernel values and term closures, and adaptive clustering for term occurrence obfuscation and storage optimization. Our design for privacy enhancement is to prevent the leakage of two critical text signals in terms of term frequency and occurrence needed for the attacks shown in the previous work and this paper.

The evaluation with two TREC datasets shows that the NDCG can be improved noticeably by replacing the exact match kernel of neural ranking with a LambdaMART tree ensemble. The obfuscation of kernel values does carry a modest relevance tradeoff for privacy. The adaptive clustering for term closures significantly reduces the storage demand with some tradeoff in relevance.

## ACKNOWLEDGMENTS

This work is supported in part by NSF IIS-1528041 and a Google faculty research award. It has used the NSF-supported resource in the Extreme Science and Engineering Discovery Environment (XSEDE) under allocation IRI190005. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

## REFERENCES

[1] Daniel Agun, Jinjin Shao, Shiyu Ji, Stefano Tessaro, and Tao Yang. 2018. Privacy and efficiency tradeoffs for multiword top k search with linear additive rank scoring. In *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 1725–1734.

[2] Jing Bai, Yi Chang, Hang Cui, Zhaohui Zheng, Gordon Sun, and Xin Li. 2008. Investigation of partial query proximity in web search. In *Proceedings of the 17th international conference on World Wide Web*. ACM, 1183–1184.

[3] Alexandra Boldyreva, Nathan Chenette, and Adam O'Neill. 2011. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In *Annual Cryptology Conference*. Springer, 578–595.

[4] Dan Boneh and Victor Shoup. 2015. A graduate course in applied cryptography. *Draft 0.2* (2015).

[5] Raphael Bost. 2016. $\sum o\varphi o\varsigma$: Forward Secure Searchable Encryption. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1143–1154.

[6] Raphael Bost and Pierre-Alain Fouque. 2017. Thwarting Leakage Abuse Attacks against Searchable Encryption – A Formal Approach and Applications to Database Padding. Cryptology ePrint Archive, Report 2017/1060. (2017). https://eprint.iacr.org/2017/1060.

[7] Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning* 11, 23-581 (2010), 81.

[8] Ning Cao, Cong Wang, Ming Li, Kui Ren, and Wenjing Lou. 2014. Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data. *IEEE Trans. Parallel Distrib. Syst.* 25, 1 (2014), 222–233.

[9] David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. 2015. Leakage-abuse attacks against searchable encryption. In *CCS'15*. ACM, 668–679.

[10] David Cash, Joseph Jaeger, Stanislaw Jarecki, Charanjit S Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. 2014. Dynamic Searchable Encryption in Very-Large Databases: Data Structures and Implementation.. In *NDSS*, Vol. 14. Citeseer, 23–26.

[11] David Cash, Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. 2013. Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries. In *CRYPTO 2013*. 353–373.

[12] David Cash and Stefano Tessaro. 2014. The Locality of Searchable Symmetric Encryption. In *EUROCRYPT 2014*. 351–368.

[13] Olivier Chapelle and Yi Chang. 2011. Yahoo! Learning to Rank Challenge Overview. *J. of Machine Learning Research* (2011), 1–24.

[14] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. 1998. Private Information Retrieval. *J. ACM* 45, 6 (Nov. 1998), 965–981.

[15] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. 2011. Searchable symmetric encryption: improved definitions and efficient constructions. *Journal of Computer Security* 19, 5 (2011), 895–934.

[16] Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. 2018. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 126–134.

[17] Van Dang. 2012. RankLib. https://sourceforge.net/p/lemur/wiki/RankLib/. (2012). Accessed: 2018-05-20.

[18] Dotan Di Castro, Liane Lewin-Eytan, Yoelle Maarek, Ran Wolff, and Eyal Zohar. 2016. Enforcing k-anonymity in web mail auditing. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM, 327–336.

[19] Tamer Elsayed, Nima Asadi, Lidan Wang, Jimmy J. Lin, and Donald Metzler. 2010. UMD and USC/ISI: TREC 2010 Web Track Experiments with Ivory. In *Proceedings of the 19th Text REtrieval Conference, TREC 2010, Gaithersburg, Maryland, USA*.

[20] Craig Gentry. 2009. Fully Homomorphic Encryption Using Ideal Lattices. In *STOC '09*. ACM, 169–178.

[21] Oded Goldreich. 2000. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, New York, NY, USA.

[22] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of CIKM'16*. ACM, 55–64.

[23] Haibo Hu, Jianliang Xu, Chushi Ren, and Byron Choi. 2011. Processing private queries over untrusted data cloud through privacy homomorphism. In *ICDE*. 601–612.

[24] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of CIKM'13*. ACM, 2333–2338.

[25] Muhammad Ibrahim and Mark Carman. 2016. Comparing Pointwise and Listwise Objective Functions for Random-Forest-Based Learning-to-Rank. *ACM Transactions on Information Systems (TOIS)* 34, 4 (2016), 20.

[26] The Ponemon Institute. 2018. The 2018 global cloud data security study. https://www2.gemalto.com/cloud-security-research. (2018). Accessed: 2018-05-01.

[27] Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. 2012. Access Pattern disclosure on Searchable Encryption: Ramification, Attack and Mitigation. In *NDSS 2012*.

[28] Geetha Jagannathan, Krishnan Pillaipakkamnatt, and Rebecca N Wright. 2009. A practical differentially private random decision tree classifier. In *2009 IEEE International Conference on Data Mining Workshops*. IEEE, 114–121.

[29] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446.

[30] Shiyu Ji, Jinjin Shao, Daniel Agun, and Tao Yang. 2018. Privacy-aware Ranking with Tree Ensembles on the Cloud. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 315–324.

[31] Shiyu Ji, Jinjin Shao, and Tao Yang. 2019. Efficient Interaction-based Neural Ranking with Locality Sensitive Hashing. In *The World Wide Web Conference (WWW '19)*. ACM, New York, NY, USA, 2858–2864.

[32] K Sparck Jones, Steve Walker, Stephen E. Robertson, et al. 2000. A probabilistic model of information retrieval: development and comparative experiments. Part 1. *Information processing & management* 36, 6 (2000), 779–808.

[33] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. 2018. GAZELLE: A Low Latency Framework for Secure Neural Network Inference. In *27th USENIX Security Symposium (USENIX Security 18)*. 1651–1669.

[34] Seny Kamara and Tarik Moataz. 2017. Boolean Searchable Symmetric Encryption with Worst-Case Sub-Linear Complexity. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 94–124.

[35] Seny Kamara, Charalampos Papamanthou, and Tom Roeder. 2012. Dynamic searchable symmetric encryption. In *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 965–976.

[36] Robert Krovetz. 2000. Viewing morphology as an inference process. *Artificial intelligence* 118, 1-2 (2000), 277–294.

[37] Tie-Yan Liu et al. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval* 3, 3 (2009), 225–331.

[38] Chris Mellor. 2018. Samsung preps for Z-SSD smackdown on Intel Optane drives. https://www.theregister.co.uk/2018/01/30/samsung_launching_zssd_attack_on_intel_optane_drives. (2018). Accessed: 2019-01-28.

[39] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[40] Muhammad Naveed, Manoj Prabhakaran, and Carl A Gunter. 2014. Dynamic searchable encryption via blind storage. In *2014 IEEE Symposium on Security and Privacy*. IEEE, 639–654.

[41] Pascal Paillier. 1999. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *EUROCRYPT '99*. 223–238.

[42] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, and Xueqi Cheng. 2017. A deep investigation of deep IR models. In *SIGIR 2017 Workshop on Neural Information Retrieval (Neu-IR'17)*.

[43] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Jingfang Xu, and Xueqi Cheng. 2017. Deeprank: A new deep architecture for relevance ranking in information

retrieval. In *Proceedings of CIKM'17*. ACM, 257–266.
[44] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.
[45] Raluca Ada Popa, Frank H. Li, and Nickolai Zeldovich. 2013. An Ideal-Security Protocol for Order-Preserving Encoding. In *SP '13*. IEEE Computer Society, 463–477.
[46] Samsung. 2018. Samsung Electronics Begins Mass Production of Industry's Largest Capacity SSD - 30.72TB - for Next-Generation Enterprise Systems. https://bit.ly/2EFKp5N. (2018). Accessed: 2019-01-28.
[47] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web*. ACM, 373–374.
[48] Lyle Smith. 2018. Intel Optane 800P NVMe SSD Review. https://www.storagereview.com/intel_optane_800p_nvme_ssd_review. (2018). Accessed: 2019-01-28.
[49] Wenhai Sun, Bing Wang, Ning Cao, Ming Li, Wenjing Lou, Y. Thomas Hou, and Hui Li. 2014. Verifiable Privacy-Preserving Multi-Keyword Text Search in the Cloud Supporting Similarity-Based Ranking. *IEEE Trans. Parallel Distrib. Syst.* 25, 11 (2014), 3025–3035.
[50] Latanya Sweeney. 2002. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10, 05 (2002), 557–570.
[51] Tao Tao and ChengXiang Zhai. 2007. An exploration of proximity measures in information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 295–302.
[52] Guofeng Wang, Chuanyi Liu, Yingfei Dong, Kim-Kwang Raymond Choo, Peiyi Han, Hezhong Pan, and Binxing Fang. 2018. Leakage Models and Inference Attacks on Searchable Encryption for Cyber-Physical Social Systems. *IEEE Access* 6 (2018), 21828–21839.
[53] Zhihua Xia, Xinhui Wang, Xingming Sun, and Qian Wang. 2016. A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. *IEEE Transactions on Parallel and Distributed Systems* 27, 2 (2016), 340–352.
[54] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 55–64.
[55] Chenyan Xiong, Zhengzhong Liu, Jamie Callan, and Tie-Yan Liu. 2018. Towards Better Text Understanding and Retrieval through Kernel Entity Salience Modeling. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '18)*. ACM, 575–584.
[56] Hamed Zamani and W Bruce Croft. 2017. Relevance-based word embedding. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 505–514.
[57] Jiashu Zhao and Jimmy Xiangji Huang. 2014. An Enhanced Context-sensitive Proximity Model for Probabilistic Information Retrieval. In *SIGIR*. 1131–1134.

# A KERNEL VALUE RECOVERY ATTACK

In this section, we describe an attack to recover the term frequency from closed soft match maps based on histogram pooling even though the exact match signals are removed. We assume that the interval $[-1, 1)$ for similarity value is divided as $[-1, b_2, \cdots, b_{R-1}, 1)$ where $-1 = b_1 \leq b_2 < b_3 < \cdots < b_{R-1} < b_R = 1$. Each kernel $K_j$ is associated with an interval $[b_j, b_{j+1})$ where $1 \leq j \leq R-1$. Note that all intervals of these kernels are disjoint and their unions are interval $[-1, 1)$. Given a term $t$, a document $d$, and a kernel vector $\vec{f}_{t,d}$ in a soft match map is $(a_1, , \cdots, a_j, \cdots, a_{R-1})^T$ and $a_R$ is not included. Each kernel value $a_j = \log K_j(t, d) = \log(\sum_{w \in d} \mathbb{1}_{b_j \leq \langle t, w \rangle < b_{j+1}})$. Thus

$$\sum_{j=1}^{R-1} \exp(a_j) = \sum_{j=1}^{R-1} (\sum_{w \in d} \mathbb{1}_{b_j \leq \langle t, w \rangle < b_{j+1}}).$$

Since the union of all the disjoint intervals is $[-1, 1)$, we can have

$$\sum_{j=1}^{R-1} \exp(a_j) = \sum_{w \in d} \mathbb{1}_{-1 \leq \langle t, w \rangle < 1}.$$

Let the length of $d$ be $n$, and term frequency of $t$ contained in $d$ is $\mathrm{TF}(t, d)$, we have $\sum_{j=1}^{R} \exp(a_j) = n$ and $\exp(a_R) = \mathrm{TF}(t, d)$. Then $\mathrm{TF}(t, d) = n - \sum_{j=1}^{R-1} \exp(a_j)$.

We also notice for any term $t'$ that is not in document $d$, $\mathrm{TF}(t', d) = 0$. But $\vec{f}_{t',d}$ is included in a soft match map because of term closure. Thus $n = \sum_{j=1}^{R-1} \exp(a'_j)$.

To launch this attack, we assume an adversary can scan the soft match map $SMM$ to obtain all keys $(t, d)$ such that $(t, d) \in SMM$, and then take the following actions: 1) For each key $(t, d)$, obtain the kernel vector $\vec{f}_{t,d} = (a_1, \cdots, a_{R-1})$ in a soft match map. Compute the sum of elements in this vector. $S_t = \sum_{j=1}^{R-1} \exp(a_j)$, where $a_j = \log K_j(t, d)$. 2) Figure out the length of this document as $n = max_{t:(t,d) \in SMM}\{S_t\}$. 3) Compute term frequency of any $t$ in $d$ as $\mathrm{TF}(t, d) = n - S_t$.

# B PROOFS

**Proof of Theorem 3.2**

Let $tf$ denote $\mathrm{TF}(t, d)$ for simplicity here.

$$|\mathrm{tf} - \exp(a_R)| = |\mathrm{tf} - \exp(\log K_R(t, d))| = |\mathrm{tf} - K_R(t, d)|$$

$$= \left| \mathrm{tf} - \sum_{w \in d} \exp(-\frac{(\langle t, w \rangle - \mu_R)^2}{2\sigma_R^2}) \right|$$

$$= \left| \mathrm{tf} - \sum_{w \in d, w = t} \exp(-\frac{(\langle t, w \rangle - 1.0)^2}{2\sigma_R^2}) - \sum_{w \in d, w \neq t} \exp(-\frac{(\langle t, w \rangle - 1.0)^2}{2\sigma_R^2}) \right|.$$

Since if $w = t$, then $\langle t, w \rangle = 1.0$, $\exp(-\frac{(\langle t, w \rangle - 1.0)^2}{2\sigma_R^2}) = \exp(0) = 1$. Thus if the length of $d$ is $n$, we have

$$\left| \mathrm{tf} - \sum_{w \in d, w = t} 1 - \sum_{w \in d, w \neq t} \exp(-\frac{(\langle t, w \rangle - 1.0)^2}{2\sigma_R^2}) \right|$$

$$= \left| \sum_{w \in d, w \neq t} \exp(-\frac{(\langle t, w \rangle - 1.0)^2}{2\sigma_R^2}) \right| \leq (n - \mathrm{tf}) \exp(-\frac{(\bar{S} - 1.0)^2}{2\sigma_R^2})$$

$$\leq n \exp(-\frac{(\bar{S} - 1.0)^2}{2\sigma_R^2}) < \epsilon.$$

**Proof of Theorem 5.2**

For each term closure, if at least one term in that closure appears in document $d$, all terms in that closure would have precomputed kernel values with $d$. For each closure $C$, there is a total of $2^{|C|} - 1$ non-empty subsets and thus there are totally $2^{|C|} - 2$ ways to replace the exact terms in closure $C$ appeared in $d$ with another subset of $C$ containing soft terms. When a server tries to guess the existence of exact term occurrence $(t, d)$, it has to locate the correct one from all $2^{|C|} - 1$ ways of forming $d$ using exact or soft terms from $C$, which is at least $2^p - 1$. When a server tries to figure out $N$ term occurrence $(t, d)$ pairs, there are $(2^p - 1)^N$ different document sets that produce the soft match maps with the same keys. The soft match maps of these $(2^p - 1)^N$ different document sets are $\varepsilon$-statistically indistinguishable because we use a subset of a term closure for replacement and the original soft match map is closed and $\varepsilon$-statistically indistinguishable.