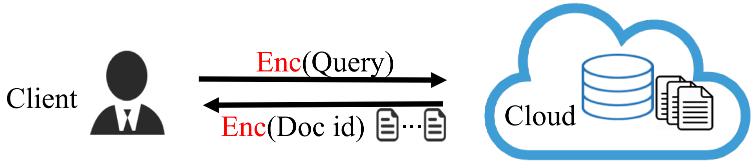
Privacy-aware Document Ranking with Neural Signals

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

Challenge for Private Search/Ranking

Client uploads encrypted documents and index, with search functionality delegating to the server, utilizing its massive storage and computing power.



Server is honest-but-curious, i.e., correctly executes protocols but observes/infers private information.

Challenges:

- Server can observe search/ranking process, and then infer private information.
- Feature leakage (e.g., term frequency) can lead to plaintext leakage.

Related Work for Private Ranking

- **Searchable Encryption**, e.g., [Cash et al. Crypto13, Curtmola et al. Crypto13] does not support ranking.
- **Leakage Abuse Attack on Search Index & Features**, e.g., [Cash et al. CCS15, Wang et al. S&P17] launches attacks with term frequency/co-occurrence.
- **Order Preserving Encryption**, e.g., [Boldyvera et al. Crypto11] does not support arithmetic operations.
- **Private Additive Ranking**, e.g., [Xia et al. TPDS16] works for small datasets only, [Agun et al. WWW18] only supports partial cloud ranking.
- **Private Tree-based Ranking**, e.g., [Bost et al. NDSS15] uses computational-heavy techniques such as Homomorphic Encryption, [Ji et al. SIGIR18] does not support neural signals.

Two Categories of Neural Ranking

Two categories of neural ranking for keyword search:

- **Representation-based** versus **interaction-based**.
- Interaction-based model outperforms in relevance benchmarks (such as $NDCG$ for ClueWeb). [Guo et al. CIKM16, Xiong et al. SIGIR17, Dai et al., WSDM18]
- Not for long/NLP queries.

Interaction-based ranking score = $f(Ker(q \otimes d))$

- q and d are embeddings for query and document.
- $\otimes$ is the interaction process, which outputs a similarity matrix containing vector similarities for all pairs of query and document term.
- Ker is the kernel computation yielding kernel vectors.
- f is the neural network computation.

Leakage in Interaction-based neural ranking

Document

m keywords

Query

n keywords

$\otimes$ *Interact*

Similarity Matrix

$m \times n$ real values

Kernel Comp. m

Kernel Vector

R real values

Term Frequency / Term Co-occurrence

Query plaintext attack, e.g., [Islam et al. NDSS12]

Proposed Solution for Private Neural Ranking

Document

m keywords

Query

n keywords

$\otimes$ *Interact*

Similarity Matrix

$m{\times}n$ real values

Kernel Comp.

Kernel Vector

R real values

~~Term frequency / cooccurrence Leakage~~

1. Pre-computed Kernel Vector

   Too much storage cost?

   Soft Match Map

2. Decomposed Kernel Vector

   Partially replace it with private tree-based model

3. Closed Soft Match Map

How Kernel Values Leak Term Frequency

$$\left\{\sum_{t\in q}\log K_1(t,d)\,,\ \sum_{t\in q}\log K_2(t,d)\,,\ldots,\ \sum_{t\in q}\log K_R(t,d)\right\}$$

$K_i(t,d)$ is the i-th kernel value on the interaction of query term t and document d. R is the number of kernels.

Decompose kernel values into two parts:

- $K_1(t,d), \ldots, K_{R-1}(t,d)$ *Soft Match Signals*
- $K_R(t,d)$ *Exact Match Signal*

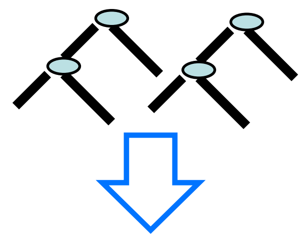
Our analysis: Term frequency of t can be well approximated by $K_R(t,d)$.

Solution for privacy-preserving: Replace the exact match signal with the private tree-based ranking signal.

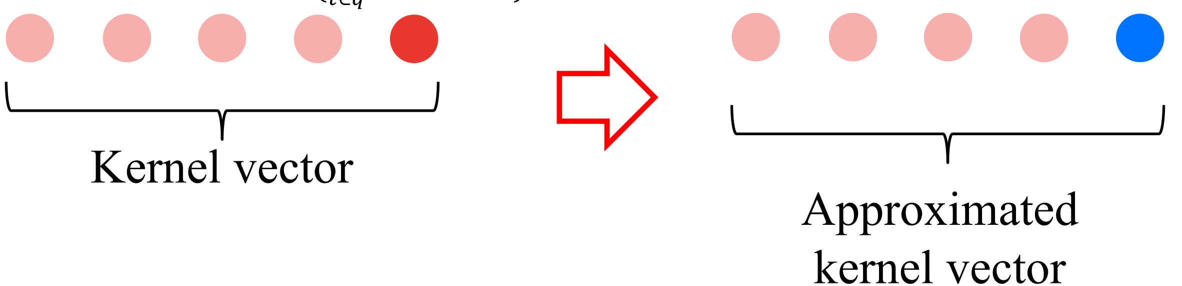
How to Approximate Exact Match Signal $K_R(t,d)$

Proposed privacy-preserving approach

1. Gather traditional word frequency and proximity features
2. Use a query-length-specific learning-to-ranking tree ensemble to compute a rank score
3. Use a private tree-based model [Ji et al., SIGIR18] to encrpt features and tree thresholds



$$\left\{\sum_{t\in q}\log K_R(t,d)\right\}$$



Closed Soft Match Map in Detail

Motivation: Limit precomputing, such that avoid to compute all possible pairs of terms and documents.
Otherwise, 1 million documents can cost ~10TB storage.
Basic idea: Precompute kernel values only for term t and document d, if ~~t appears in d~~ t is soft-relevant to d. *Soft match map*: key is (t,d) and value is the kernel vector

Challenge: How to define soft-relevant that is privacy-preserving? E.g., not leak term occurrence, which can facilitate plaintext attacks.

Address privacy concern with Closed Soft Match Map:
For two terms t_0 and t_1, if 1) (t_0, d) is in Soft Match Map; 2) t_0 and t_1 are similar, then (t_1, d) is also in Soft Match Map.
Next step: Cluster all terms into the similarity closures.

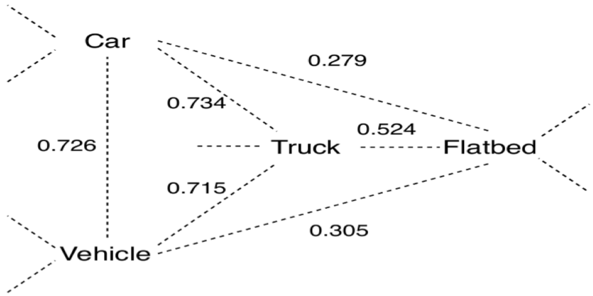
Options to Build Term Closures based on Fixed or Adaptive Similarity Threshold

If terms t_0 and t_1 in a τ-similar term closure, $sim(t_0, t_1) \geq \tau$.

Fixed-threshold Clustering: Apply a uniform τ for all closures.

Weakness: Closures can include too many (or too few) terms, which incurs huge storage cost (or privacy leakage).

Adaptive Clustering: Given a closure size limit p, apply a series of increasing thresholds: $\tau_1 < \tau_2 \ldots < \tau_m$, to gradually expand all term closures that are of size below p.



Clustering with fixed threshold:

C1: {Car, Truck, Vehicle, Flatbed, ...}

Adaptive clustering:

C1: {Car, Truck, Vehicle},
C2: {FlatBed, ...}

Privacy Property of Closed Soft Match Map

Objective: Given a closed soft match map, a server adversary cannot learn term frequency/occurrence of the dataset.

Property Sketch: Given a dataset D, with N key-value pairs in a closed soft match map of D, and closure size $\geq p$, there exist at least $(2^p - 1)^N$ different datasets D' such that their soft match maps have the same key, and values that are ε-statistically indistinguishable

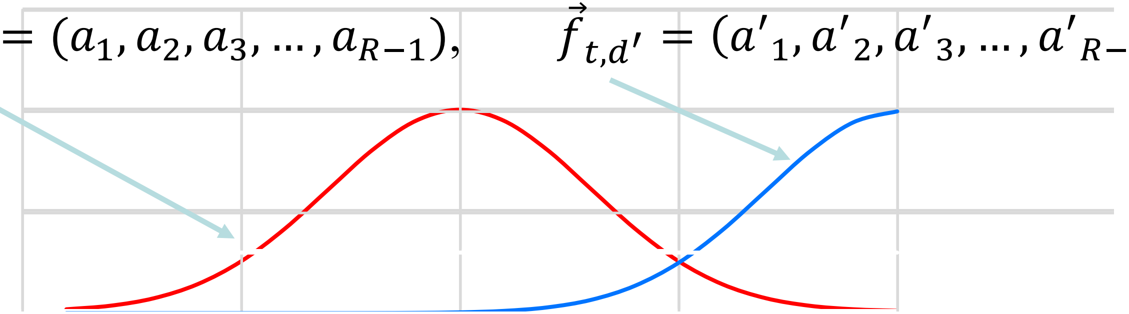
Takeaway: These $(2^p - 1)^N$ different datasets have different term frequencies and co-occurrences, while their soft match maps are very similar.

Thus, the cloud server is unlikely to recover the correct dataset.

More on Indistinguishable Kernel Values

Kernel values of a term *t* in documents *d* and *d′* are:

$\vec{f}_{t,d} = (a_1, a_2, a_3, \ldots, a_{R-1}),\qquad \vec{f}_{t,d'} = (a'_1, a'_2, a'_3, \ldots, a'_{R-1}).$



***ε*-statistically indistinguishable kernel values:**

$$\varepsilon \geq Statistical\ Dist.\left(\vec{f}_{t,d}, \vec{f}_{t,d'}\right) = \frac{1}{2}\sum_{i=1}^{R-1} |a_i - a'_i|$$

i.e., an adversary can successfully differentiate d and d' with probability at most $\frac{1}{2} + \varepsilon$.

Takeaway:

$$\downarrow \varepsilon \xrightarrow{yields} \downarrow Prob(\text{successfully differentiate between } d \text{ and } d')$$

Minimize Statistical Distance of Kernel Values

A method to minimize $\boldsymbol{Statistical\ Dist.}(\vec{f}_{t,d}, \vec{f}_{t,d'})$:

For the j-th soft kernel value in the kernel value vector, it is **obfuscated** as:

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

where r is a privacy parameter, t is a term, d is a document, and $K_j(t,d)$ is the output from the j-th kernel function.

Trade-off between Privacy and Ranking Accuracy:

$\uparrow r \xrightarrow{\text{yields}} \downarrow Statistical\ Dist. \xrightarrow{\text{yields}} \uparrow$ Privacy Guarantee

$\xrightarrow{\text{yields}} \downarrow$ Effectiveness of Soft Match Signals

Evaluation Setup

- ✓ **Leverage:** 1) Privacy-aware search/feature access[Agun et al. WWW18] 2) Private tree ensemble model [Shiyu et al. SIGIR18] with CPM encrypted features & Query length specific training.
- ✓ **Datasets:** [Robust04] contains ~0.5 million documents with 250 queries. [ClueWeb09-Cat-B] contains ~50 million documents with 150 queries from Web 09-11.

- **Evaluation Objectives:**
  1. Can approximated kernel vectors with private tree ensemble signals rank well?
  2. Can kernel value obfuscation preserve the ranking accuracy?
  3. How effective are two different methods of clustering term closures?

Evaluation on Approx. Exact Match Signal

	ClueWeb09-Cat-B			Robuts04		
Model	NDCG@1	NDCG@3	NDCG@10	NDCG@1	NDCG@3	NDCG@10
LambdaMART	0.2893	0.2828	0.2827	0.5181	0.4610	0.4044
DRMM	0.2586	0.2659	0.2634	0.5049	0.4872	0.4528
KNRM	0.2663	0.2739	0.2681	0.4983	0.4812	04527
C-KNRM	0.3155	0.3124	0.3085	0.5373	0.4875	0.4586
C-KNRM*	0.2884	0.2927	0.2870	0.5007	0.4702	0.4510
C-KNRM*/T	0.3175	0.3122	0.3218	0.5404	0.5006	0.4657

C-KNRM is CONV-KNRM [Dai et al. WSDM18]
C-KNRM* is a version of CONV-KNRM without bigram-bigram interaction
C-KNRM*/T is C-KNRM* while using a LambdaMART tree ensemble to replace the exact match signal of kernel vectors.

Takeaway: Tree signal intergration for neural kernel vectors perform well and even boost ranking performance.

Choices on Tree-based Ranking Feature

	ClueWeb09-Cat-B			Robuts04		
Model	NDCG@1	NDCG@3	NDCG@10	NDCG@1	NDCG@3	NDCG@10
C-KNRM	0.3155	0.3124	0.3085	0.5373	0.4875	0.4586
C-KNRM*	0.2884	0.2927	0.2870	0.5007	0.4702	0.4510
C-KNRM*/T1	0.3175	0.3122	0.3218	0.5404	0.5006	0.4657
C-KNRM*/T2	0.3038	0.2998	0.2933	0.5149	0.4827	0.4535

T2 only includes term frequency features. T1 includes T2 plus proximity and page quality features.

Takeaway: Different features for tree-based models can have significant impact on ranking performance when approx. exact match signal.

Effectiveness of Kernel Value Obfuscation

	ClueWeb09-Cat-B			Robuts04		
Model	NDCG@1	NDCG@3	NDCG@10	NDCG@1	NDCG@3	NDCG@10
C-KNRM	0.3155	0.3124	0.3085	0.5373	0.4875	0.4586
C-KNRM*	0.2884	0.2927	0.2870	0.5007	0.4702	0.4510
C-KNRM*/TO No Obfuscation	0.3175	0.3122	0.3218	0.5404	0.5006	0.4657
C-KNRM*/TO r = 5	0.3178	0.3067	0.3100	0.5306	0.4987	0.4613
C-KNRM*/TO r = 10	0.3121	0.3097	0.3100	0.5221	0.4980	0.4623

C-KNRM*/TO is C-KNRM* while using the tree-approximated kernel vectors and kernel value obfuscation

Takeaway: Kernel value obfuscation results in small degradation (~1.6%) on ranking performance, when r = 10.

Evaluation on Term Clustering Methods

	ClueWeb09-Cat-B			Robuts04		
Clustering Method	NDCG@1	NDCG@3	NDCG@10	NDCG@1	NDCG@3	NDCG@10
C-KNRM	0.3155	0.3124	0.3085	0.5373	0.4875	0.4586
Fixed Threshold $\tau = 0.3$	0.3136	0.3078	0.3091	0.5225	0.4974	0.4621
Fixed Threshold $\tau = 0.7$	0.3064	0.3048	0.3104	0.4886	0.4644	0.4169
Adaptive $\tau = 0.3$	0.3052	0.3069	0.3120	0.5127	0.4892	0.4582
Adaptive $\tau = 0.7$	0.3067	0.3012	0.3060	0.4899	0.4608	0.4090

C-KNRM*/TOC is C-KNRM* while using the tree-approximated kernel vectors, kernel value obfuscation, and closed soft map

Takeaway: 1) Clustering threshold choices have an impact on relevance. 2) Adaptive thereshold is competitive to fixed threshold while saving up to ~40 storage cost. (Details in Table.3 of this paper.)

Concluding Remarks

- **Contribution:** a privacy-aware neural ranking
- **Evaluation results** with two datasets

1. NDCG can be improved by replacing the exact match kernel of neural ranking with a tree ensemble.
2. The obfuscation of kernel values does carry a modest relevance trade-off for privacy.
3. The adaptive clustering for term closures significantly reduces the storage demand with some trade-off in relevance.