Privacy-aware Ranking with Tree Ensembles on the Cloud

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

ABSTRACT

Tree-based ensembles are widely used for document ranking but supporting such a method efficiently under a privacy-preserving constraint on the cloud is an open research problem. The main challenge is that letting the cloud server perform ranking computation may unsafely reveal privacy-sensitive information. To address privacy with tree-based server-side ranking, this paper proposes to reduce the learning-to-rank model dependence on composite features as a trade-off, and develops comparison-preserving mapping to hide feature values and tree thresholds. To justify the above approach, the presented analysis shows that a decision tree with simplifiable composite features can be transformed into another tree using raw features without increasing the training accuracy loss. This paper analyzes the privacy properties of the proposed scheme, and compares the relevance of gradient boosting regression trees, LambdaMART, and random forests using raw features for several test data sets under the privacy consideration, and assesses the competitiveness of a hybrid model based on these algorithms.

ACM Reference Format:
Shiyu Ji, Jinjin Shao, Daniel Agun, Tao Yang Department of Computer Science, University of California at Santa Barbara. 2018. Privacy-aware Ranking with Tree Ensembles on the Cloud. In *SIGIR '18: The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, July 8–12, 2018, Ann Arbor, MI, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3209978.3210022

1 INTRODUCTION

As sensitive information is increasingly stored on the cloud, privacy concerns have been a critical factor for users to adopt cloud-based information services [53]. In offering a service, a server can observe the client-initiated query processing flow, and reason about client's data even it is encrypted. Unencrypted feature values and statistic information about index can lead to privacy attacks [10]. To address the emerging need of privacy on the cloud, searchable encryption [11–14, 18, 28, 32–34, 50] has been proposed to identify documents that match a user query from the encrypted index, but has not considered ranking. Recent research has proposed privacy-aware ranking with linear additive scoring in [2, 9, 51, 56], but the work of [9, 51, 56] can only be applicable for small datasets while the solution in [2] is targeted for modest-sized datasets since its ranking partially relies on client computation and client-server communication can become a bottleneck for a very large dataset. It is an open problem to develop a server-side privacy-aware ranking solution for a large dataset, especially using nonlinear tree ensembles.

Ensemble trees derived with machine learning techniques such as Gradient Boosted Regression Trees (GBRT)[22, 40], LambdaMART [8] are popular for feature vector classification and have been proven to be very effective, outperforming a linear additive formula for ranking with web-scale document collections in handling dynamic queries. For example, in the Yahoo! learning-to-rank challenge [15], all winners have used some forms of tree ensembles. Recently, random forests by bagging with GBRT and LambdaMART [26] have been shown to be competitive.

The main challenge in developing privacy-aware techniques for server-side nonlinear tree-based ranking is that such computation involves not only query-dependent arithmetic calculations in organizing features, but also numerical comparison to navigate decision trees. While fully homomorphic encryption [23] can support server-side addition and multiplication over encrypted data without decrypting such data, such a scheme is not computationally scalable. For example, a fast implementation of fully homomorphic encryption [25] takes more than 320 seconds for adding 1024 encrypted integers. Partially homomorphic encryption can improve time efficiency in a limited degree, and still has other limitations, for example, results computed are not comparable at the server side [42]. Order-preserving encryption techniques [1, 5, 43, 44] let a server compare the encrypted results but do not support arithmetic computation on encrypted numbers.

The contribution of this paper is an efficient privacy-aware server-side ranking scheme with tree ensembles and to the best of our knowledge, this is the first effort to address such a problem by exploiting a relevance and privacy trade-off. Our key idea is to reduce the dependence of tree ensembles on *composite* features that require dynamic query-based calculation as a trade-off (e.g. BM25), and rely more on *raw* features to train tree ensembles (e.g. document term frequency). To justify the above approach, we show that decision trees with certain composite features can be transformed into ones with raw features, without increasing regression or entropy-based loss. Based on such a setting, we propose comparison-preserving mapping that hides document feature values while supporting minimum and maximum feature composition. With participation of more raw features, the model training becomes hybrid with a query length specific selection of the base algorithms and feature mixing.

The rest of the paper is organized as follows. Section 2 defines the problems and Section 3 reviews the related work. Section 4 gives an overview of our design considerations and approach. Section 5 analyzes the change of learning accuracy loss when transforming a tree with composite features. Section 6 presents a mapping to hide document features and tree thresholds, and provides the leakage profile and privacy properties. Section 7 presents the evaluation results. Section 8 concludes our paper. All proofs are listed in Appendix A.

SIGIR '18, July 8–12, 2018, Ann Arbor, MI, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5657-2/18/07…$15.00
https://doi.org/10.1145/3209978.3210022

2 PROBLEM DEFINITION

Privacy assumptions and requirements. 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. We adopt the previous work on searchable encryption and assume that a server is able to conduct privacy-aware query processing that identifies encrypted documents matching a client query [2, 11, 18, 32] and assume that a server can access a set of encrypted *raw* features for each matched document. Notice certain information is still leaked during matching, for example, query word statistical information after launching many queries and partial leakage of the posting length of query words. Details on the leakage profile can be found in [2].

How to encrypt these features by the client while the server can conduct tree-ensemble based ranking without knowing these values is the problem we focus on. We emphasize the proposed solution is privacy aware to preserve privacy as much as possible and the server does not learn non-trivial information about documents. Certain information is still leaked, for example, relative rank order of documents, and tree structure. We should provide a leakage profile of the proposed scheme.

Raw and composite features. A rank feature is called *raw* if it is explicitly stored in the index and it is called *composite* if it is computed dynamically based on raw features. Often a composite feature computation is query dependent. An example is BM25 or TFIDF [31] which is the summation of term frequency based raw features and it is query-dependent, and thus cannot be precomputed and stored in the index.

In general a document that matches a query is represented by a raw feature vector $(f_1, \cdots, f_m)$ where m may depend on the length of a query. The standard learning-to-rank process for a tree-ensemble model such as GBRT [36], LambdaMART [8], and random forests [26, 27] uses document feature vectors of a fixed length, independent of the query length. The server has to compute composite features such as BM25 and proximity from raw features and then form a feature vector of a fixed length including some of raw features. Examples of raw features directly used for tree ensemble derivation include query specific features such as document-query click through rate and document specific features such as freshness and document quality.

Tree ensembles for document ranking. A learning algorithm for tree ensembles produces a set of decision trees T_i, $1 \le i \le n$. Given document d represented by a feature vector with a fixed length after computing composite features aggregating some of raw features, each tree in a learned ensemble gives a score prediction $T_i(d)$. The final ranking score is defined as $\sum_{i=1}^{n} \alpha_i \cdot T_i(d)$, where α_i is a weight associated to the i-th tree. In GBRT and LambdaMART, weight α_i is learned based on the boosting effect of trees from T_1, $\cdots$, T_{i-1} while in a random forest approach [26, 27], an ensemble with multiple trees is learned first given a random selection of features, and then multiple ensembles are additively combined. Thus the weights for combining multiple trees depend on how trees are ensembled and each decision tree has the following characteristics:

- Each non-leaf node has a predicate in the form $f \ge t$, where f is a feature value and t is a trained threshold. When this node is reached during ranking, if the predicate is satisfied by the feature value f of a document, the right subtree is further visited. Otherwise the left subtree is visited.
- Each leaf has a score value. When the leaf is finally reached during ranking, this value is used as a score contribution $T_i(d)$ from this i-th tree.

3 RELATED WORK

GBRT uses a point-based cost with the squared prediction error as the loss function. LambdaMART [8] is list-based by updating the parameters using on the NDCG metric in revising the optimization target of next iteration. Once the optimization parameters are determined at each learning iteration, LambdaMART derives a regression tree using the point-based squared error as the loss function. A random forest scheme employs many decision trees to predict a value using a bagging or boosting manner [35]. Each decision tree in a random forest is derived using a squared regression loss function or based on information gain theory.

Searchable symmetric encryption (SSE) has been studied in [11, 12, 18, 34]. Curtmola et al. [18] formalized the notion of SSE as a Structured Encryption [16], and Cash et al. [12] gave the first provably secure (up to some presumed leakage) SSE scheme that supports conjunctive search over encrypted inverted index, which has been recently extended to disjunctive multiword queries also [32]. The work in [9, 41, 51, 56, 60] studies TFIDF-based additive ranking with private similarity computation through matrix transformation and as shown in [2], such a scheme does not scale well. More efficient method is proposed in [2] for a modest sized dataset while ranking is conducted partially at the server side. Our work will be focused on *complete* server-side ranking with nonlinear tree ensembles.

Private decision tree research for classification tasks is conducted in [6, 29, 55] which only considers to give a classification score for a given feature vector. It does not consider ranking, and thus does not support query-dependent feature composition or server-side ordering among scored vectors. For making tree computation private, research in [6, 55] uses computationally-heavy cryptographic techniques including homomorphic encryption[42], garbled circuit [58] and oblivious transfer [47]. The cost of computing a score for a single document through a small number of trees is at a level of seconds, and will not be scalable for handling a query with many matched documents. There is also a line of work perturbing the feature values to protect user privacy [29, 37] for classification tasks. Our method is orthogonal to these since our goal is to preserve the exact ranking model.

Order preserving encryption (OPE) has been studied in database and cryptography communities [1, 5, 60]. As the core of OPE, order preserving mapping (OPM) has been used to convert one set of numbers into another set while preserving the order [5, 54, 60] and the decryption is not supported. OPE cannot support arithmetic computation on encrypted numbers. Zhang et al. [60] proposed an OPM that can support addition, i.e., the sum of two values' OPM results is also an OPM result of the sum of these two values. However this construction often yields a mapping which is almost linear and thus not private enough as an attack can be formed to discover this approximated linear mapping function.

4 DESIGN CONSIDERATION AND OVERVIEW

Privacy-aware feature selection and composition. Since it is too expensive to use homomorphic encryption to support server-side computation within a reasonable response time, our first design strategy is to restrict the type of arithmetic operations supported at the server side in feature composition to seek a trade-off in ranking accuracy.

Particularly, we will support the minimum and maximum based composition among raw features. An example of using min/max feature composition is to leverage the minimum distance among two query words that appear in a document [52, 61]. This design does not intend to support other types of feature composition operators including multiplications and addition. For static query-independent coefficients involved in a ranking formula, we may embed them in raw features stored in the index if possible. In general, to compensate the loss of composite signals, we propose to directly use raw features if they can serve as partial relevance signals without feature composition. That may lead to a possible degradation of ranking accuracy, and in next section, we will provide an analytic result to justify this approach.

We illustrate how to represent BM25 [39, 48] using raw features. Given query Q, the original BM25 formula is $\Sigma_{t_i \in Q}(k_1 + 1)\frac{tf(t_i,d)(k_1+1)}{K+tf(t_i,d)} \log \frac{N}{df_i} \frac{tf(t_i,Q)(k_3+1)}{k_3+tf(t_i,Q)}$ where $tf(t_i, d)$ is the occurrence count of term t_i in document d, df_i is the number of documents that term t_i appears, and N is the number of documents in the whole collection. $K = k_1((1 - b) + b \cdot dl/avdl)$ where dl is the document length and $avdl$ is the average document length for the whole collection. k_1, k_3, and b are constants. This formula is modified in [38] by dropping the query-dependent coefficient $\frac{tf(t_i,Q)(k_3+1)}{k_3+tf(t_i,Q)}$ for better accuracy and this modification fits better for our scheme which does not support server-side dynamic multiplication. Then we use $(k_1 + 1)\frac{tf(t_i,d)(k_1+1)}{K+tf(t_i,d)} \log \frac{N}{df_i}$ as individual raw features to participate in the tree buildup.

For proximity features, traditional inverted index stores word positions explicitly [3] and online ranking computes composite proximity features based on word positions through arithmetic calculation, e.g. position difference in computing query word spans. We can avoid that by following the previous work that uses word-pair or n-gram based features as a substitute [4, 21, 61]. For example, $\sum_{q_1,q_2 \in Q} \frac{1}{dist(q_1,q_2,d)^2}$ where $dist(t_1, t_2, d)$ is the minimum distance of these two terms within document d. We can let raw features $\frac{1}{dist(q_1,q_2,d)^2}$ directly participate in the tree learning, or use $\max_{q_1,q_2 \in Q} \frac{1}{dist(q_1,q_2,d)^2}$ [4] as one supported composite feature.

Query length specific training with hybrid tree ensemble model selection. Since the total number of individual raw features (e.g. introduced for BM25 or TF-IDF) depends on the query length, we have to partition the training dataset and derive an ensemble model separately for each query length. The average number of query words is known to vary between 2 and 3 based on several studies from search engine logs [30, 49]. In practice, the length of queries to be processed in a document search system can be limited by a constant (e.g. 10) and queries with an excessive number of query words may be trimmed. Thus only a relatively small number of tree ensembles needs to be trained for different query lengths.

Different algorithms behave differently in terms of validation or test accuracy. Training a tree ensemble model for each query length also yields an opportunity that we can use a different learning-to-rank algorithm for each different query length. Also raw features and corresponding min and/or max features can be complementary, letting them co-exist in the training process leads to extra choices of the model selection. Thus a hybrid approach can train a different model for a different setting by choosing the smallest validation error given options of multiple tree algorithms and choices of raw and composite features.

Hiding feature values and tree thresholds. To hide feature values and node thresholds used to compare these values in decision trees, one option is to use an order preserving mapping function called OPM from [5, 54, 60]. Let F denote the set of all the possible feature values that can be compared with certain thresholds in decision trees, and T denote the set of such thresholds. Then an OPM function ensures that $\forall v_1, v_2 \in (F \cup T), v_1 > v_2 \leftrightarrow OPM(v_1) > OPM(v_2)$, and $v_1 = v_2 \leftrightarrow OPM(v_1) = OPM(v_2)$.

Notice that OPM allows all of features in F to be compared with each other which is not necessary in our targeted problem as we only need support a feature in F be comparable with a threshold in T. With this in mind, we propose a feature encoding method called comparison-preserving mapping that is sufficient to preserve the correctness of decision branching in tree ensembles, but reveals much less information to the server compared to OPM.

Finally, we also need to hide the leaf values of trees as much as possible. Motivated by [2], we adopt a random offset method by adding a random offset R_i to every leaf value. Suppose for the i-th tree, a leaf value is chosen as $T_i(d)$. When computing the ranking score of N trees, the new ranking score will be $\sum_{i=1}^{N} \alpha_i(T_i(d)+R_i) = \sum_{i=1}^{N} \alpha_i T_i(d) + \sum_{i=1}^{N} \alpha_i R_i$, where α_i is the weight of the i-th tree. Note that $\sum_{i=1}^{N} \alpha_i T_i(d)$ is the true ranking score, and $\sum_{i=1}^{N} \alpha_i R_i$ is a fixed offset. Hence the relative rank order among documents is preserved. The above random offset method leaks leaf rank value difference within each tree. Given the server knows the relative rank score order of matched documents, such a leakage is considered to be acceptable.

5 TREE ACCURACY WITH RAW FEATURES

While we expect some relevance degradation by restricting the type of composite feature computation, we are investigating this impact by assessing the tree accuracy measured by the training loss function for trees that use raw features instead of composite features.

Given a composition function $g()$ using k raw features $f_1, \cdots, f_k$, we call this function *inequality-simplifiable* if we fix any $k - 1$ features of these k raw features with constants in the inequality $g(f_1, f_2, \cdots, f_k) \geq t$, where t is a constant threshold, then this inequality can be transformed into an equivalent form $f_i \geq t'$ where t' is another constant threshold, and f_i is the one of these

k raw features which is not fixed with a constant. A composite feature is inequality-simplifiable if it uses an inequality-simplifiable composition function. For example, let $g(f_1, f_2) = 2f_1 + 3f_2$, it is inequality-simplifiable because $g(c_1, f_2) \geq t$ and $g(f_1, c_2) \geq t$, where c_1 and c_2 are constants, can be transformed as $f_2 \geq (t-2c_1)/3$ and $f_1 \geq (t-3c_2)/2$, respectively. Other examples of inequality-simplifiable functions include: $f_1 \log f_2$ and $\frac{1}{1+e^{-f_1-f_2}}$. The composition features used in our evaluation are all inequality-simplifiable. We believe a large percentage of composite features used in the previous work and in practice fall into such a category.

Now we show that each decision tree that uses *inequality-simplifiable* composite features can be transformed into another tree using raw features only without training loss degradation when the loss function is the squared regression error or entropy gain. To illustrate this concept, Fig. 1 gives an example of transforming a tree using a sum-based composite feature into another tree using raw features only. There are 8 training instances marked as 3 white circles with relevance label 0 and 5 black circles with relevance label 1. The new tree can separate white and black circles as accurate as the old tree and the regression error of both trees is 0. The composite feature example $f_1 + f_2$ is inequality-simplifiable.

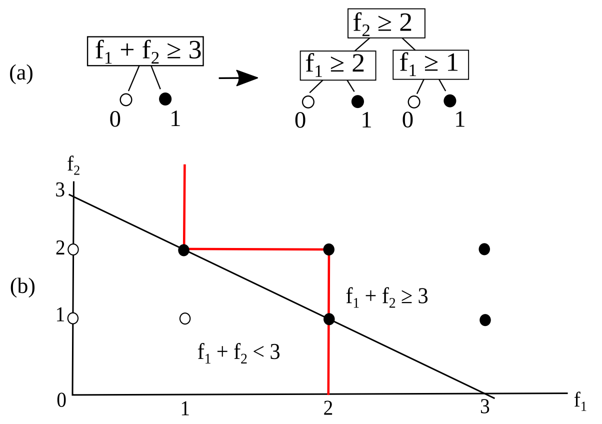

Figure 1: (a) A tree using composite feature $f_1 + f_2$ is transformed into another tree using raw features f_1 and f_2. (b) The regression lines corresponding to two trees separate 8 training instances in white and black circles.

A tree algorithm typically uses the squared regression error or the entropy-based information gain[45, 46] as the loss metric in tree buildup. Following the formulation [22, 26, 40], we list the loss metric definition as follows. Given a training instance $t_i = (x_i, y_i)$ associated with a query where x_i is a feature vector and y_i is a judgment label. Let $Leaves(A)$ be the set of leaf nodes in the tree A. Given a leaf v in the tree, denote by $D(v)$ the set of all the training instances which choose the path from the root to the leaf v based on their feature values. Let $D(A)$ be the set of all the training instances covered by any of the leaves of the (sub)tree A. The squared error loss of tree A is:

$$\sum_{v \in Leaves(A)} \sum_{t_i \in D(v)} (y_i - \ell(v))^2$$

where $\ell(v)$ denotes the leaf value of v. The information gain of the tree A is

$$-\sum_{j=1}^{n} P(j) \cdot \log P(j) - E(A)$$

where n is the number of target classes, $P(j)$ is the probability that a training instance is in the j-th class, and conditional entropy $E(A)$ is defined as:

$$E(A) = \sum_{v \in Leaves(A)} \frac{|D(v)|}{|D(A)|} \cdot \left(-\sum_{j=1}^{n} P(j|v) \cdot \log P(j|v) \right),$$

where $P(j|v)$ denotes the conditional probability that a training instance falling into leaf v is in the j-th class.

The following result can be shown that transforming tree A to tree B with a decomposition of training instance subsets does not increase the loss. Notice that the training instance set of a leaf in tree A is decomposed into disjoint subsets in tree B and each of these subsets is exactly covered by one leaf in Tree B.

Lemma 5.1. *Given two decision trees A and B that satisfy $D(A) = D(B)$ and $\forall u \in Leaves(A), \exists V \subset Leaves(B)$, $D(u) = \cup_{v \in V} D(v)$. Then the loss of Tree B is no larger than that of tree A when squared error or entropy-based information gain is used as the loss function.*

Lemma 5.2 shows that we can gradually simplify a composition feature of a tree through tree transformation, which leads to Theorem 5.3.

Lemma 5.2. *A decision tree using an inequality-simplifiable composite feature based on k raw features can be transformed into another tree using a composite feature based on k-1 raw features with no larger squared error and no smaller information gain.*

Theorem 5.3. *A decision tree that uses inequality-simplifiable composite features can be transformed into another tree using only raw features with no larger squared error and no smaller information gain.*

The above theorem can be applied to every tree in a multiple-tree scheme such as gradient boosting regression trees, LambdaMART trees, and random forests. It should be emphasized that the accuracy of decision trees without respect to training loss is only a proximity to the final relevance results. If using the transformation discussed in the proof of the above lemmas and theorem, the depth of a transformed tree becomes much larger, which can lead to an overfitting situation, and thus we still follow a common practice to limit the tree size so that each tree acts as a "weak" classifier.

6 FEATURE AND THRESHOLD ENCODING

This section gives an encoding method called comparison preserving mapping to hide feature values and tree thresholds, and we also discuss the leakage profile and privacy properties of our method.

6.1 Comparison Preserving Mapping

In deriving the CPM, we associate all distinct raw feature values in a document collection with comparable thresholds in decision trees as one group and call it *comparable group*. Then we can partition feature values and thresholds into many disjoint groups. Notice all raw feature values compared using the maximum or minimum composition should be considered in the same comparable group because all raw features are compared with the associated tree thresholds indirectly through the maximum or minimum operator. For each comparable group of raw feature values and associated thresholds in tree ensembles, we derive a comparison preserving mapping. Then we use such a mapping to encode the feature values and thresholds of each group.

Let all thresholds of each comparable group be: $T = [t_1, t_2, \cdots, t_r]$ in a sorted order so that $t_i \le t_{i+1}$. F is the set of all associated feature value comparable with these thresholds, Then define a comparison-preserving mapping function called CPM as

$$\forall t_i \in T, \mathrm{CPM}(t_i) = i;$$

$$\forall f \in F, \mathrm{CPM}(f) = \begin{cases} 0 & \forall t_i \in T, f < t_i, \\ argmax_i\{t_i \in T \wedge t_i \le f\} & \text{otherwise.} \end{cases}$$

Example. Figure 2 illustrates the CPM procedures for 3 trees using 3 thresholds with one comparable group. Namely $T = \{0.5, 3, 5\}$ and $F = \{0.3, 0.8, 1.5, 2.5, 3.8, 5.1\}$ for features f_1 and f_2 of documents d_1, d_2, and d_3. Then the CPM of these thresholds is $\mathrm{CPM}(0.5) = 1$, $\mathrm{CPM}(3) = 2$, $\mathrm{CPM}(5) = 3$. Six feature values are encoded as one of these 3 values or 0.

The above example illustrates that CPM hides feature values and thresholds as 6 feature values and 3 thresholds are mapped to 4 values {0, 1, 2, 3} without revealing real values. Also 4 threshold and feature values { 0.5, 0.8, 1.5, 2.5} are all mapped to the same new value 1 and the server is not able to distinguish them. We enumerate the privacy properties in next subsection.

In terms of ensuring the correctness of tree computation for ranking, the following result shows that CPM retains the correctness of threshold comparison in decision trees for all raw features and min/max-based composite features.

Theorem 6.1.

$$\forall f \in F,\ \forall t \in T,\ f \ge t \leftrightarrow CPM(f) \ge CPM(t).$$

$\forall f_i \in F, \forall t_j \in T$, we have

$$\max(f_1, \cdots, f_k) \ge t_j \leftrightarrow \max(CPM(f_1), \cdots, CPM(f_k)) \ge CPM(t_j)$$

and

$$\min(f_1, \cdots, f_k) \ge t_j \leftrightarrow \min(CPM(f_1), \cdots, CPM(f_k)) \ge CPM(t_j).$$

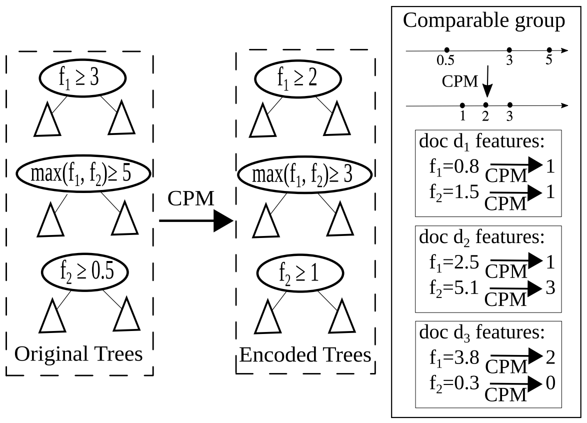

Figure 2: An example of CPM in a tree ensemble.

Comparison with OPM. Table 1 gives a detailed comparison of mapping properties between CPM and OPM. A key difference is that while OPM strictly preserves the order among features and thresholds, CPM only guarantees an encoded feature value is comparable with an encoded tree node threshold. The server may not know the order between the CPM-encoded feature values when these values are the same. Thus CPM reveals less information than OPM.

Properties ($\forall v_1, v_2 \in F$)	M = CPM	$M = OPM$
$v_1 > v_2 \rightarrow M(v_1) > M(v_2)$	False	True
$v_1 > v_2 \leftarrow M(v_1) > M(v_2)$	True	True
$v_1 = v_2 \rightarrow M(v_1) = M(v_2)$	True	True
$v_1 = v_2 \leftarrow M(v_1) = M(v_2)$	False	True

Table 1: Mapping difference between CPM and OPM.

Storage space requirement with CPM. The space cost for storing features with CPM is efficient because each feature value can be represented with $\log_2 N$ bits where N is the number of distinct thresholds used for one feature value group. A typical number of trees involved can be up to tens of thousands while each tree contains tens or few hundred nodes. Thus the total number of distinct thresholds involved for each feature group is in a level of thousands or millions. Then $\log_2 N$ would typically lead to 2 or 3 bytes for storing each feature value.

6.2 Leakage profile and privacy properties

First we summarize the leakage profile of our scheme as follows. Namely the following information can be revealed to a server that hosts the encoded features and tree ensembles.

- Tree ensemble structure information including 1) the number of trees, 2) the topology of each tree, 3) the membership of each comparable feature and threshold group. 4) The score value difference between every two leaves in a tree.
- The order information among distinct thresholds and feature values after CPM encoding within each comparable group. Namely, for any two thresholds t_1, t_2 in each group, if $\mathrm{CPM}(t_1) > \mathrm{CPM}(t_2)$, then the server can infer $t_1 > t_2$. For any feature f_1 and threshold t_1, $\mathrm{CPM}(f_1) \ge \mathrm{CPM}(t_1)$, then the server can infer $f_1 \ge t_1$.
- The order information between feature values in the same comparable group if CPM of these features are different. Namely $\mathrm{CPM}(f_1) > \mathrm{CPM}(f_2)$, then the server can infer $f_1 > f_2$. But if $\mathrm{CPM}(f_1) = \mathrm{CPM}(f_2)$, then the server cannot be sure that $f_1 = f_2$.
- The number of distinct thresholds, and their distribution information at each comparable feature and threshold group. The distribution of distinct feature values is leaked when for any two distinct feature values f_1, f_2, there always exists a threshold t in the comparable group such that $\mathrm{CPM}(f_1) < \mathrm{CPM}(t) \le \mathrm{CPM}(f_2)$.

Next we characterize the key privacy properties of our scheme, namely what information is protected.

- A server cannot well approximate the actual values of feature values or thresholds, their difference, and their ratios. Theorem 6.2 gives a formal description of this property.
- For feature values and thresholds associated with different comparable groups, the server cannot compare them.

Collection	1	2	3	4	5	Total
Robust04	11	70	140	25	4	250
Robust05	1	19	24	5	1	50
Web09-12	64	70	52	14	0	200
MQ09	98	294	232	53	9	686

Table 2: Number of queries of different query lengths

- For any two features values $v_i < v_j$ in a comparable group, if there exist two thresholds t_1, t_2 associated with this group such that $t_1 \le v_i < v_j \le t_2$ and there is no other threshold in the group between t_1 and t_2, then we must have $\mathrm{CPM}(v_i) = \mathrm{CPM}(v_j)$ and the probability that the server correctly figures out v_j is larger is at most 0.5, i.e., no better than random guess.

Now we formally characterize the first privacy property listed above. Let $\mathrm{CPM}(F, T)$ denote the encoded threshold and feature value group of a dataset. Let $v_i \in F \cup T$ denote a feature value or threshold that can be identified as i-th number by the server and $\forall v_i, v_i \in [a, b]$, but the server does not know about this domain.

A server can only use $CPM(F, T)$ to derive an approximation of value $v_i \in F \cup T$ or its difference/ratio to other values. The three properties in Theorem 6.2 below show that in order to approximate within a specified error bound, this server has to correctly distinguish the one from an infinite number of choices with the same CPM encoding and different domains, but exceeding this bound, which is highly unlikely.

Theorem 6.2. *The following three properties are true.*

- **P1:** *Let $A_v(i, CPM(F, T))$ denote a server algorithm that can approximate the original i-th value in $F \cup T$ within error ϵ using $CPM(F, T)$. Namely, for any original i-th feature value $v_i \in F \cup T$, $|A_v(i, CPM(F, T)) - v_i| < \epsilon$. For such algorithm A_v, there exist an infinite number of datasets, where each is denoted as $(\tilde{F}, \tilde{T})$, such that $CPM(\tilde{F}, \tilde{T}) = CPM(F, T)$, and $|A_v(i, CPM(\tilde{F}, \tilde{T})) - \tilde{v}_i| > \epsilon$.*
- **P2:** *Let $A_d(i, j, CPM(F, T))$ denote a server algorithm that can approximate difference $v_i - v_j$ within error ϵ for any i-th and j-th values: $v_i, v_j \in F \cup T$. Without loss of generality, assume $v_i > v_j$. Namely, $|A_d(i, j, CPM(F, T)) - (v_i - v_j)| < \epsilon$. For such algorithm A_d, there exist an infinite number of datasets, where each is denoted as $(\tilde{F}, \tilde{T})$, such that $CPM(\tilde{F}, \tilde{T}) = CPM(F, T)$, and $|A_d(i, j, CPM(\tilde{F}, \tilde{T})) - (\tilde{v}_i - \tilde{v}_j)| > \epsilon$.*
- **P3:** *Assume all feature values and thresholds in a comparable group are nonnegative and $v_{\min}$ is the smallest among these nonzero values. Let $A_r(i, CPM(F, T))$ denote a server algorithm that can use $CPM(F, T)$ to approximate ratio $v_i/v_{\min}$ for any $v_i > v_{\min}$ within any error ϵ such that $0 < \epsilon < v_i/v_{\min} - 1$, $|A_r(i, CPM(F, T)) - \frac{v_i}{v_{\min}}| < \epsilon$. For such algorithm A_r, there exist an infinite number of datasets, where each is denoted as $(\tilde{F}, \tilde{T})$, such that $CPM(\tilde{F}, \tilde{T}) = CPM(F, T)$, and $|A_r(i, CPM(\tilde{F}, \tilde{T})) - \frac{\tilde{v}_i}{\tilde{v}_{\min}}| > \epsilon$.*

For Properties P1 and P2, the domain $[a, b]$ of feature and threshold values can include negative values if needed and these properties hold if $a \ge 0$ without any change. Property P3 assumes $a \ge 0$. To extend Property P3 when $a < 0$, we can consider v_{min} as the smallest absolute non-zero value and then can show that a server cannot approximate $|\frac{v_i}{v_{min}}|$ within error ϵ such that $0 < \epsilon < |\frac{v_i}{v_{\min}}| - 1$.

7 EVALUATION

Datasets. We use the following datasets with four TREC test collections. 1) Robust04 uses TREC Disks 4 and 5 (excluding Congressional Records) with 0.5M news articles and 250 topic queries from TREC Robust track 2004. 2) Robust05 uses TREC AQUAINT collection with 1M news articles and 50 queries from TREC Robust track 2005. 3) Web09-12 uses Clueweb09 Category B with 50M webpages. The 200 queries are from the TREC Web Tracks 2009-2012. 4) MQ09 also uses Clueweb09 Category B with 686 queries from Million Query Track 2009. Since we will conduct query length specific training, Table 2 lists the length of TREC queries for each dataset after stop word removal.

Implementation and model training. We follow the work of [2, 13, 32] that can retrieve the matched results and encrypted features to build inverted index for the datasets TREC 4&5, AQUAINT and Clueweb09-Cat-B. For model training, we used RankLib 2.5 [19], which contains LambdaMART, GBRT and random forests (RF), following 5-fold cross validation, and the hybrid model is an extension of them by selecting the one with the smallest validation error under different feature group arrangements. Following other previous work in relevance evaluation(e.g. [7, 20, 24, 59]), we report NDCG@20 score of the baseline algorithms and our approach. To identify an ideal model, we have varied the number of trees size from 100 to 1000 until no better improvement is found, and also varied the tree size from 4 to 32 leaves.

Features. We form 4 candidate groups of features for training.

- **G0**: BM25 for query words that appear in the title, and BM25 for query words that appear in the body. The minimum, maximum, and average of the squared min distance reciprocal of query word pairs in the title or in the body. For Clueweb09, extra features include PageRank and a binary flag indicating whether a document is from Wikipedia.
- **G1**: All features from Group G0 except that BM25 and word-pair proximity are replaced by their individual raw scores.
- **G2**: All features from Group G1 plus the maximum and minimum proximity score in title or body section.
- **G3**: All features from Group G2 after excluding those individual raw scores that compose the proximity score.

Tree algorithms. Our evaluation compares the performance of following approaches: 1) Each of LambdaMART, GBRT and random forest methods is trained for all queries with G0 group features; As LambdaMART outperforms GBRT for the Clueweb09 collection and we use LambdaMART as a base algorithm for random forests where each tree is configured to use 30% feature sampling rate and each bag uses 1 tree; 2) Linear additive ranking based on AdaRank [57] using G0 to show performance difference compared to nonlinear tree ensembles. 3) Each tree model trained for each query length using one of G1, G2, and G3. 4) A hybrid model that selects a configuration from one of the tree algorithms and G1, G2, and G3 feature groups discussed below with best validation performance.

As the training queries in Table 2 are not enough for some of query lengths for training and testing, we mitigate this query shortage by adopting two strategies: 1) For single word queries, since the composite BM25 or proximity feature is the same as an individual raw feature, we use the best tree model trained by queries of all lengths. 2) For query length above 4, there are only few queries available after stop word removal. Thus for each query with length larger than 4 in Robust04/05 and Web09-12, we only take the first

4 query words and group them with 4 word queries. Namely all queries with 4 or more words are ranked by a model trained using the grouped 4-word queries .

A comparison of overall relevance. Table 3 lists the average NDCG@20 results with 5-fold validation for queries in Table 2. This table shows that 1) The overall performance of the hybrid model is constantly among the best or close to the best across all datasets compared to other 3 algorithms. The hybrid model achieves the best performance in Robust04, Web09-12, and MQ09. For Robust04, RF method does the best, but its advantage is not significant (less than 0.2%) compared to the hybrid. 2) Compared to the baseline of 3 tree algorithms using feature group G0 with no privacy constraints, the performance of our hybrid model is competitive with a small relevance degradation. For Robust04 and Robust05, the hybrid model is better by 1%-5.5%. For Web09-12 and MQ09, the hybrid model is 1%-3.3% worse. Thus the relevance trade-off of replacing composite features with raw features is reasonable. From Column 5 of this table, linear additive ranking based on AdaRank underperforms the tree algorithms as expected, which confirms the positive value of making tree ranking private.

Relevance with raw and min/max features for different query lengths. To explain why the hybrid model gives more stable results than other 3 algorithms, Table 4 gives the query length specific NDCG@20 results in 4 sections of columns. All the results are within confidence interval ±0.01 with p-value < 0.05. A boldface number for each algorithm column section represents the best performance among feature groups G1, G2, and G3 under the same algorithm. We observe that there is no dominating method and no dominating feature groups when varying QL from 1 to 5, and the hybrid model selects the best configuration during validation which typically leads to the highest NDCG test score except a few cases.

For all single word queries (Rows marked with $QL = 1$), composite features have the same values as raw features. As we use the queries in all different lengths to train, the testing results for single word queries are identical among all groups with the fixed model under the same algorithm. LambdaMART performs better for Robust05 and Web09-12 while the random forest method does better for Robust04 and MQ09. The hybrid model selects the best configuration among them.

When $QL \geq 2$, the hybrid model typically retains the best performance among all configurations, except a few cases. For example in dealing with MQ-09 and $QL = 4$, the hybrid follows GBRT with G2 having the smallest validation error while the random forest method with G2 actually has the higher NDCG test score. Similarly the configuration that the hybrid model selects does not deliver the highest NDCG for Robust05 with $QL = 4$. For those cases the hybrid model does not lead to the highest NDCG, the performance gap with the best is relatively small.

Consistency of relevance scores with the previous work. We examine the NDCG@20 scores achieved in the previous work in using the same datasets. For Web09-12, the NDCG@20 score in [20, 59] has reported 0.2273 and 0.2461 with neural network signals and 0.1939 using traditional IR signals. NDCG@20 in [7] is 0.2331 using features similar to G0, and 100 queries from TREC 2010 and 2011 while we use 200 queries. In comparison, LambdaMART delivers 0.2235 in our evaluation without using neural signals. For MQ09, NDCG@20 in [7] is 0.2428 using 450 queries while in our evaluation, LambdaMART delivers 0.2603 using 686 queries. For Robust04, NDCG@20 is 0.3794 in [20] without neural signals and is 0.4509 with neural signals while it is 0.3982 in [59]. Random forests can reach 0.4114 in our evaluation. Overall speaking, our scores are on a par with those in the previous work without using neural network features, even there is a difference in data preprocessing and feature choices. Our future work is to integrate or extend our privacy-aware scheme with the use of neural features.

Space cost for CPM. For two test datasets that use ClubWeb09 Category B, we have used up to 2000 trees and 32 leaves and identified up to about 40,000 thresholds. The growth of this number is slow as we increase the tree size or the number of trees. The number of bits required for CPM is 16 and thus each of document features and tree thresholds is represented by two bytes. Thus the space cost of CPM in representing features and thresholds is small.

8 CONCLUSION

The main contribution of this paper is a privacy-aware scheme for server-side document ranking using tree ensembles with reasonable relevance. The proposed scheme with comparison-preserving mapping can scale for large datasets since a server does not involve time-consuming heavyweight cryptography or additional client involvement after receiving encrypted search words from a client. While restricting the computation complexity of feature composition represents a trade-off of relevancy for privacy, we exploit characteristics of inequality simplifiable feature composition and demonstrate that decision trees that replace raw features with such composition features can still be competitive. The evaluation results show that the hybrid model trained for each query length can perform competitively as the best algorithm in each configuration setting with a small relevance degradation compared to a traditional tree algorithm.

This paper uses traditional document features for ranking and future work is to consider the latest development in neural network based ranking [20, 24, 59]. The proposed feature encoding can support minimum and maximum operators and an extension for other types of composition can be considered in the future.

Acknowledgments. We thank Stefano Tessaro for his critical advice and valuable help, and the anonymous referees for their thorough comments. This work is supported in part by NSF IIS-1528041. 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] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order preserving encryption for numeric data. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 563–574. ACM, 2004.
[2] D. Agun, J. Shao, S. Ji, S. Tessaro, and T. Yang. Privacy and efficiency tradeoffs for multiword top k search with linear additive rank scoring. In *WWW*, 2018.
[3] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval (2nd Edition)*. Addison Wesley, 2011.
[4] J. Bai, Y. Chang, H. Cui, Z. Zheng, G. Sun, and X. Li. Investigation of partial query proximity in web search. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 1183–1184, 2008.
[5] A. Boldyreva, N. Chenette, and A. O'Neill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In *Annual Cryptology Conference*, pages 578–595. Springer, 2011.

Collection	λ-MART G0	GBRT G0	RF G0	AdaRank G0	λ-MART G1	λ-MART G2	λ-MART G3	RF G1	RF G2	RF G3	Hybrid
Robust04	0.3936	0.4025	**0.4114**	0.3756	0.3791	0.3843	0.3774	**0.3980**	0.3912	0.3870	0.3975
Robust05	0.2765	0.2778	**0.2945**	0.2541	0.2694	0.2702	0.2627	0.2736	0.2827	0.2795	**0.2928**
Web09-12	**0.2235**	0.1906	0.2100	0.1853	0.2047	0.2063	0.2048	0.2040	0.2042	0.2040	**0.2160**
MQ09	**0.2603**	0.2419	0.2395	0.2390	0.2552	0.2524	0.2446	0.2234	0.2349	0.2289	**0.2573**

Table 3: NDCG@20 of different methods for datasets from Table 2.

		λ-MART				GBRT				RF				
Collection	QL	G0	G1	G2	G3	G0	G1	G2	G3	G0	G1	G2	G3	Hybrid
Robust04	1	0.5155	0.5155	0.5155	0.5155	0.5220	0.5220	0.5220	0.5220	0.5225	**0.5225**	**0.5225**	**0.5225**	**0.5225**
	2	0.4120	0.4163	**0.4196**	**0.4196**	0.4071	0.4087	**0.4140**	**0.4140**	0.4248	**0.4242**	0.4222	0.4222	**0.4242**
	3	0.3877	0.3702	**0.3784**	0.3655	0.4046	**0.3792**	0.3668	0.3595	0.4028	**0.3904**	0.3800	0.3730	**0.3904**
	4	0.3312	**0.2809**	0.2779	0.2803	0.3361	0.2638	**0.3100**	0.3042	0.3730	**0.3125**	0.3098	0.3073	0.3073
Robust05	1	0.3355	**0.3355**	**0.3355**	**0.3355**	0.2244	0.2244	0.2244	0.2244	0.1468	0.1468	0.1468	0.1468	**0.3355**
	2	0.3217	**0.3221**	0.3198	0.3198	0.3278	0.3210	**0.3241**	**0.3241**	0.3514	**0.3624**	0.3501	0.3501	**0.3624**
	3	0.2491	0.2400	**0.2436**	0.2371	0.2505	**0.2480**	0.2418	0.2427	0.2652	0.2126	**0.2432**	0.2381	**0.2480**
	4	0.2335	0.2088	**0.2090**	0.1724	0.2294	0.2021	**0.2243**	0.2082	0.2483	**0.2547**	0.2431	0.2365	0.2431
Web09-12	1	0.2191	**0.2191**	**0.2191**	**0.2191**	0.1804	0.1804	0.1804	0.1804	0.1909	0.1909	0.1468	0.1909	**0.2191**
	2	0.2204	0.2211	**0.2213**	**0.2213**	0.1934	**0.1913**	0.1768	0.1768	0.2451	**0.2476**	0.2471	0.2471	0.2395
	3	0.2201	0.1670	0.1603	**0.1670**	0.1862	**0.1228**	0.1170	0.1153	0.1664	0.1474	**0.1463**	0.1454	**0.1670**
	4	0.2715	0.1974	**0.2439**	0.1974	0.2396	0.1612	**0.2461**	0.2296	0.2834	0.2561	**0.2658**	**0.2658**	**0.2658**
MQ09	1	0.1866	0.1866	0.1866	0.1866	0.1850	0.1850	0.1850	0.1850	0.1883	**0.1883**	**0.1883**	**0.1883**	**0.1883**
	2	0.2786	0.2710	**0.2712**	**0.2712**	0.2506	0.2319	**0.2457**	**0.2457**	0.2609	0.2448	**0.2612**	**0.2612**	**0.2712**
	3	0.2706	**0.2767**	0.2683	0.2470	0.2534	0.2169	**0.2185**	0.2154	0.2378	0.2192	**0.2284**	0.2151	**0.2767**
	4	0.2782	**0.2281**	0.2280	0.2195	0.2754	0.1922	**0.2296**	0.2264	0.2496	0.2188	**0.2369**	0.2193	0.2296
	5	0.0913	0.0910	**0.0913**	**0.0913**	0.0810	0.0388	0.0843	**0.0913**	0.0826	**0.0388**	**0.0388**	0.0264	**0.0913**

Table 4: NDCG@20 for different query lengths.

[6] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser. Machine learning classification over encrypted data. In *NDSS*, 2015.

[7] L. Boytsov and A. Belova. Evaluating learning-to-rank methods in the web track adhoc task. In *TREC*, 2011.

[8] C. J. Burges. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81, 2010.

[9] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou. Privacy-preserving multi-keyword ranked search over encrypted cloud data. *IEEE Transactions on parallel and distributed systems*, 25(1):222–233, 2014.

[10] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart. Leakage-abuse attacks against searchable encryption. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 668–679. ACM, 2015.

[11] D. Cash, J. Jaeger, S. Jarecki, C. S. Jutla, H. Krawczyk, M.-C. Rosu, and M. Steiner. Dynamic searchable encryption in very-large databases: Data structures and implementation. In *NDSS*, volume 14, pages 23–26. Citeseer, 2014.

[12] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Roşu, and M. Steiner. Highly-scalable searchable symmetric encryption with support for boolean queries. In *Advances in Cryptology–CRYPTO 2013*, pages 353–373. Springer, 2013.

[13] D. Cash and S. Tessaro. The locality of searchable symmetric encryption. In *EUROCRYPT 2014*, pages 351–368, 2014.

[14] Y.-C. Chang and M. Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In *ACNS'05*, pages 442–455, 2005.

[15] O. Chapelle and Y. Chang. Yahoo! Learning to Rank Challenge Overview. *J. of Machine Learning Research*, pages 1–24, 2011.

[16] M. Chase and S. Kamara. Structured encryption and controlled disclosure. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 577–594. Springer, 2010.

[17] T. M. Cover and J. A. Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[18] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. *Journal of Computer Security*, 19(5):895–934, 2011.

[19] V. Dang. Ranklib-a library of ranking algorithms.

[20] M. Dehghani, H. Zamani, A. Severyn, J. Kamps, and W. B. Croft. Neural ranking models with weak supervision. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 65–74. ACM, 2017.

[21] T. Elsayed, N. Asadi, L. Wang, J. J. Lin, and D. Metzler. UMD and USC/ISI: TREC 2010 web track experiments with ivory. In *Proceedings of The Nineteenth Text REtrieval Conference, TREC 2010, Gaithersburg, Maryland, USA, November 16-19, 2010*, 2010.

[22] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

[23] C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC '09*, pages 169–178. ACM, 2009.

[24] J. Guo, Y. Fan, Q. Ai, and W. B. Croft. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 55–64. ACM, 2016.

[25] S. Halevi and V. Shoup. Bootstrapping for helib. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 641–670. Springer, 2015.

[26] M. Ibrahim. Scalability and performance of random forest based learning-to-rank for information retrieval. In *ACM SIGIR Forum*, volume 51, pages 73–74. ACM, 2017.

[27] M. Ibrahim and M. Carman. Comparing pointwise and listwise objective functions for random-forest-based learning-to-rank. *ACM Transactions on Information Systems (TOIS)*, 34(4):20, 2016.

[28] M. S. Islam, M. Kuzu, and M. Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *NDSS 2012*, 2012.

[29] G. Jagannathan, K. Pillaipakkamnatt, and R. N. Wright. A practical differentially private random decision tree classifier. In *Data Mining Workshops, 2009. ICDMW'09. IEEE International Conference on*, pages 114–121. IEEE, 2009.

[30] B. J. Jansen and A. Spink. How are we searching the world wide web? a comparison of nine search engine transaction logs. *Information processing & management*, 42(1):248–263, 2006.

[31] K. S. Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: development and comparative experiments. *Inf. Process. Manage.*, 36(6):779–808, Nov. 2000.

[32] S. Kamara and T. Moataz. Boolean searchable symmetric encryption with worst-case sub-linear complexity. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 94–124. Springer, 2017.

[33] S. Kamara and C. Papamanthou. Parallel and dynamic searchable symmetric encryption. In *FC 2013*, pages 258–274, 2013.

[34] S. Kamara, C. Papamanthou, and T. Roeder. Dynamic searchable symmetric encryption. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 965–976. ACM, 2012.

[35] A. Liaw, M. Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

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

[37] X. Liu, Q. Li, T. Li, and D. Chen. Differentially private classification with decision tree ensemble. *Applied Soft Computing*, 2017.

[38] Y. Lv and C. Zhai. Lower-bounding term frequency normalization. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 7–16. ACM, 2011.

[39] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval.* Cambridge Unviersity Press, 2008.

[40] A. Mohan, Z. Chen, and K. Q. Weinberger. Web-search ranking with initialized gradient boosted regression trees. In *Yahoo! Learning to Rank Challenge*, pages 77–89, 2011.

[41] C. Örencik and E. Savaş. An efficient privacy-preserving multi-keyword search over encrypted cloud data with ranking. *Distributed and Parallel Databases*, 32(1):119–160, 2014.

[42] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT '99*, pages 223–238, 1999.

[43] R. A. Popa, F. H. Li, and N. Zeldovich. An ideal-security protocol for order-preserving encoding. In *SP '13*, pages 463–477. IEEE Computer Society, 2013.

[44] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan. Cryptdb: Protecting confidentiality with encrypted query processing. In *SOSP '11*, pages 85–100. ACM, 2011.

[45] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[46] J. R. Quinlan. *C4. 5: programs for machine learning.* Elsevier, 2014.

[47] M. O. Rabin. How to exchange secrets with oblivious transfer. *IACR Cryptology ePrint Archive*, 2005:187, 2005.

[48] S. E. Robertson, S. Walker, M. Beaulieu, M. Gatford, and A. Payne. Okapi at trec-4. In *Proceedings of the fourth text retrieval conference*, volume 500, pages 73–97, 1996.

[49] C. Silverstein, H. Marais, M. Henzinger, and M. Moricz. Analysis of a very large web search engine query log. In *ACm SIGIR Forum*, volume 33, pages 6–12. ACM, 1999.

[50] D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *SP '00*. IEEE Computer Society, 2000.

[51] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y. T. Hou, and H. Li. Verifiable privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking. *IEEE Transactions on Parallel and Distributed Systems*, 25(11):3025–3035, 2014.

[52] T. Tao and C. Zhai. 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*, SIGIR '07, pages 295–302. ACM, 2007.

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

[54] C. Wang, N. Cao, K. Ren, and W. Lou. Enabling secure and efficient ranked keyword search over outsourced cloud data. *IEEE Transactions on parallel and distributed systems*, 23(8):1467–1479, 2012.

[55] D. J. Wu, T. Feng, M. Naehrig, and K. Lauter. Privately evaluating decision trees and random forests. *Proceedings on Privacy Enhancing Technologies*, 2016(4):335–355, 2016.

[56] Z. Xia, X. Wang, X. Sun, and Q. Wang. A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. *IEEE Transactions on Parallel and Distributed Systems*, 27(2):340–352, 2016.

[57] J. Xu and H. Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 391–398. ACM, 2007.

[58] A. C.-C. Yao. How to generate and exchange secrets. In *Foundations of Computer Science, 1986., 27th Annual Symposium on*, pages 162–167. IEEE, 1986.

[59] H. Zamani and W. B. Croft. Relevance-based word embedding. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 505–514. ACM, 2017.

[60] W. Zhang, Y. Lin, S. Xiao, J. Wu, and S. Zhou. Privacy preserving ranked multi-keyword search for multiple data owners in cloud computing. *IEEE Transactions on Computers*, 65(5):1566–1577, 2016.

[61] J. Zhao and J. X. Huang. An enhanced context-sensitive proximity model for probabilistic information retrieval. In *SIGIR*, pages 1131–1134, 2014.

A PROOFS

Proof of Lemma 5.1 : Part I. When the loss function of single tree generation is based on the squared error of training instances, the squared error of each leaf v is minimized when the value of each leaf is chosen as the mean value

$$\mu_v = \frac{\sum_{t_i \in D(v)} y_i}{|D(v)|} = \arg\min_u \sum_{t_i \in D(v)} (y_i - u)^2.$$

The squared error loss of entire tree A is:

$$\begin{aligned}
&\sum_{v \in Leaves(A)} \sum_{t_i \in D(v)} (y_i - \mu_v)^2 \\
&= \sum_{v \in Leaves(A)} \Big(\sum_{t_i \in D(v)} y_i^2 - 2 \sum_{t_i \in D(v)} y_i \mu_v + |D(v)| \mu_v^2 \Big) \\
&= \sum_{v \in Leaves(A)} \Big(\sum_{t_i \in D(v)} y_i^2 - |D(v)| \mu_v^2 \Big) \\
&= \sum_{t_i \in D(A)} y_i^2 - \sum_{v \in Leaves(A)} \frac{(\sum_{t_i \in D(v)} y_i)^2}{|D(v)|}.
\end{aligned}$$

With all positive constants s_1, s_2, c_1, and c_2, knowing $(c_1 s_2 - c_2 s_1)^2 \geq 0$, this inequality is true: $\frac{(s_1+s_2)^2}{(c_1+c_2)} \leq \frac{s_1^2}{c_1} + \frac{s_2^2}{c_2}$. This inequality can be generalized for all positive constants s_i and c_i as

$$\frac{(\sum_{i=1}^n s_i)^2}{\sum_{i=1}^n c_i} \leq \sum_{i=1}^n \frac{s_i^2}{c_i}.$$

Given $D(A) = D(B)$ and $\forall u \in Leaves(A), \exists V, V \subset Leaves(B) \wedge D(u) = \cup_{v \in V} D(v)$, and a training instance can only uniquely choose one leaf in each tree,

$$\begin{aligned}
&\sum_{t_i \in D(A)} y_i^2 - \sum_{v \in Leaves(A)} \frac{(\sum_{t_i \in D(v)} y_i)^2}{|D(v)|} \\
\geq &\sum_{t_i \in D(B)} y_i^2 - \sum_{v \in Leaves(B)} \frac{(\sum_{t_i \in D(v)} y_i)^2}{|D(v)|}.
\end{aligned}$$

Part II. Now we analyze when the entropy-based information gain is used as the loss function. Note that by definition the training instance sets of leaves in tree A are disjoint for different leaves. That is true also for tree B. Based on the condition of this lemma, for any leave u of A, the training instance set $D(A)$ is decomposed into a disjoint subset collection called V in tree B. V exactly covers instances in $D(u)$. Namely $D(u) = \cup_{v \in V} D(v) = D(V)$. Following [17], the information gain is always nonnegative. Namely

$$E(V) - \sum_{w \in V} \frac{|D(w)|}{|D(V)|} \cdot E(w) \geq 0.$$

Then the conditional entropy of tree A can be compared with the conditional entropy of tree B as:

$$\begin{aligned}
E(A) &= \sum_{v \in Leaves(A) \backslash u} \frac{|D(v)|}{|D(A)|} \cdot E(v) + \frac{|D(u)|}{|D(A)|} \cdot E(u) \\
&\geq \sum_{v \in Leaves(A) \backslash u} \frac{|D(v)|}{|D(A)|} \cdot E(v) + \frac{|D(u)|}{|D(A)|} \cdot \sum_{w \in V} \frac{|D(w)|}{|D(u)|} \cdot E(w) \\
&= \sum_{v \in Leaves(B) \backslash V} \frac{|D(v)|}{|D(B)|} \cdot E(v) + \sum_{w \in V} \frac{|D(w)|}{|D(B)|} \cdot E(w) \\
&= E(B).
\end{aligned}$$

Hence the information gain with tree B is no less than that with tree A. □

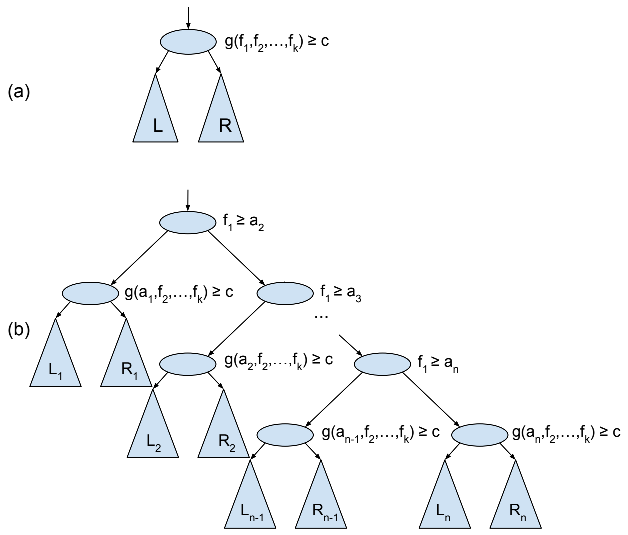

Figure 3: (a) A tree using an inequality-simplifiable composite feature. (b) Transformed tree.

Proof of Lemma 5.2: Assume that a tree called A contains a node called C using a composite feature in inequality $g(f_1, \cdots, f_k) \geq c$. We transform this tree shown in Figure 3(a) to Tree B shown in Figure 3(b).

where node C is replaced by a right-deep structure constructed as follows.

- We partition the training instances of $D(C)$ based on the values of their raw feature f_1 and assume there are n sorted distinct values for f_1: $a_1 < a_2 < \cdots < a_n$. The nodes in this right-deep structure use sorted inequalities as $f_1 \geq a_2$, $f_1 \geq a_3$, until $f_1 \geq a_n$. The left child of condition node $f_1 \geq a_{i+1}$ uses inequality $g(a_i, f_2, \cdots, f_k) \geq c$ composed of $k-1$ raw features. The left and right subtrees of this new node are the mirrored left and right subtrees of C in tree A.
- The training instance subset relationship between tree A and tree B satisfies:

$$D(L) = \cup_{i=1}^{n} D(L_i), \quad D(R) = \cup_{i=1}^{n} D(R_i).$$

Using Lemma 5.1, the transformed tree has no larger squared error and no smaller information gain compared to the original one using composite features. □

Proof of Theorem 5.3: By applying Lemma 5.2 iteratively, any tree that uses composite features can be transformed into a tree which only relies on raw features and has no larger squared error and no smaller information gain. □

Proof of Theorem 6.1. Notice t_k is the k-th sorted threshold in $T_{\geq}$. Given $f \in F$, when $f \geq t_k$, $\text{CPM}(f) = argmax_i\{1 \leq i \leq r \wedge t_i \leq f\}$, then $\text{CPM}(f) \geq k$. Since $\text{CPM}(t_k) = k$, we have $\text{CPM}(f) \geq \text{CPM}(t_k)$.

When $\text{CPM}(f) \geq \text{CPM}(t_k) = k$, $argmax_i\{1 \leq i \leq r \wedge t_i \leq f\} \geq k$. then $\forall i, 1 \leq i \leq k, t_i \leq f$. Thus $f \geq t_k$.

With the above result, we can infer that the server can perform the minimum and maximum composite operations correctly on CPM-encoded features

$$\text{CPM}(\max(f_1, \cdots, f_k)) = \max(\text{CPM}(f_1), \cdots, \text{CPM}(f_k)))$$
$$\text{CPM}(\min(f_1, \cdots, f_k)) = \min(\text{CPM}(f_1), \cdots, \text{CPM}(f_k)))$$

Namely our mapping does not affect the correctness of the min/max composite features used in decision trees. □

Proof of Theorem 6.2.

For Property P1, we find another dataset $\tilde{F}, \tilde{T}$ by adding a number $c > 2\epsilon$ to each feature value in F and threshold in T. Namely $\tilde{v}_i = v_i + c$ and $\tilde{t}_j = t_j + c$ for any $v_i \in F$ and $t_j \in T$. Then we have the identical CPM encodings: $\text{CPM}(\tilde{F}) = \text{CPM}(F), \text{CPM}(\tilde{T}) = \text{CPM}(T)$ except that the domain of $\tilde{F}$ and $\tilde{T}$ is $[a + c, b + c]$.

Since the approximation algorithm derives the same result from the same CPM encodings, $A_v(i, \text{CPM}(\tilde{F}, \tilde{T})) = A_v(i, \text{CPM}(F, T))$. Hence $A_v(i, \text{CPM}(\tilde{F}, \tilde{T})) - \tilde{v}_i = A_v(i, \text{CPM}(F, T)) - (v_i + c) \in (-c - \epsilon, -c + \epsilon)$ for any i. Since $c > 2\epsilon$, the absolute error for this dataset must be larger than ϵ. Note that since there are infinite choices of c as long as $c > 2\epsilon$, the possible choices of $(\tilde{F}, \tilde{T})$ are also infinite.

For Property P2, we find dataset $\tilde{F}, \tilde{T}$ by multiplying a number $\alpha > 1 + 2\epsilon/\delta$ to each feature value in F and threshold in T, where δ is the minimum absolute difference between any two different feature values. Thus $\tilde{v}_i = \alpha \cdot v_i$ and $\tilde{t}_j = \alpha \cdot t_j$ for any $v_i \in F$ and $t_j \in T$. The domain of $\tilde{F}$ and $\tilde{T}$ is $[\alpha \cdot a, \alpha \cdot b]$.

Then $\tilde{v}_i - \tilde{v}_j = \alpha(v_i - v_j)$. Without loss of generality, assume $v_i > v_j$. Then

$$\begin{aligned} A_d(i, j, \text{CPM}(\tilde{F}, \tilde{T})) - (\tilde{v}_i - \tilde{v}_j) &= A_d(i, j, \text{CPM}(F, T)) - \alpha(v_i - v_j) \\ &\in ((1 - \alpha)(v_i - v_j) - \epsilon, (1 - \alpha)(v_i - v_j) + \epsilon). \end{aligned}$$

Since $(1 - \alpha)(v_i - v_j) < -2\epsilon \frac{v_i - v_j}{\delta} \leq -2\epsilon$, the absolute approximation error with $\tilde{F}$ and $\tilde{T}$ must be larger than ϵ. Note that since there are infinite choices of α as long as $\alpha > 1 + 2\epsilon/\delta$, the possible choices of $(\tilde{F}, \tilde{T})$ are also infinite.

For Property P3, we construct dataset $\tilde{F}, \tilde{T}$ as follows: 1) all zero feature/threshold values in F, T are not changed in $\tilde{F}, \tilde{T}$; 2) all non-zero feature/threshold values are subtracted by number c such that $v_{\min} > c > 2v_{\min}/3$. The domain of $\tilde{F}$ and $\tilde{T}$ is $[max(0, a - c), b - c]$, which is still nonnegative. $\text{CPM}(\tilde{F}, \tilde{T}) = \text{CPM}(F, T)$. Hence

$$\begin{aligned} &A_r(i, \text{CPM}(\tilde{F}, \tilde{T})) - \tilde{v}_i/\tilde{v}_{\min} \\ =&A_r(i, \text{CPM}(F, T)) - (v_i - c)/(v_{\min} - c) \\ <&A_r(i, \text{CPM}(F, T)) - v_i/v_{\min} + v_i/v_{\min} - \frac{3v_i - 2v_{\min}}{v_{\min}} \\ =&A_r(i, \text{CPM}(F, T)) - v_i/v_{\min} - \frac{2v_i - 2v_{\min}}{v_{\min}} \\ \in&(-\epsilon - \frac{2v_i - 2v_{\min}}{v_{\min}}, \epsilon - \frac{2v_i - 2v_{\min}}{v_{\min}}). \end{aligned}$$

Since $\epsilon < v_i/v_{\min} - 1$, we have

$$\epsilon - \frac{2v_i - 2v_{\min}}{v_{\min}} < -\frac{v_i - v_{\min}}{v_{\min}} < -\epsilon.$$

Thus the absolute error with $\tilde{F}$ and $\tilde{T}$ must be larger than ϵ. Note that since there are infinite choices of c as long as $v_{\min} > c > 2v_{\min}/3$, the possible choices of $(\tilde{F}, \tilde{T})$ are also infinite. □