Privacy Preserving Similarity Search for Online Advertising

Patrick Zhang

MIT PRIMES - June 2020

Abstract

Online advertisements are an essential component of the business model of many online platforms and often the main source of revenue. Ads are most effective when relevant ads are shown to the proper audience, which requires detailed personal information to be collected on web users. This is accomplished using Similarity Search algorithms which match user interests to ads. However, current techniques make no attempt at providing privacy to the end users. We propose a novel approach to the similarity search problem for selecting ads by designing a private interactive protocol that provides user privacy without compromising ad selection accuracy. Our protocol matches the accuracy of existing Similarity Search algorithms and is experimentally evaluated using real-world data.

1 Introduction

Online advertisement is an essential aspect of online media applications. Advertisement is the primary source of revenue for the majority of websites as advertisers pay websites to display ads to its users. Advertisers invest into displaying ads with the expectation that the exposure will increase their customer count. Thus, the audience is an important factor to advertisers trying to maximum customer count. Ads should be shown to users who display interest in the same types of products. Targeted advertising is already implemented in practice using a variety of similarity search algorithms to match up users to ads. However, in order for these algorithms to perform accurately, vast amounts of information on users of websites must be collected through various means such as browsing history and previously clicked ads. The lack of privacy regarding the collection of personal information has been a constantly growing concern with the general public, creating a movement towards regulation on data collection.

We aim to create a system for privacy preserving similarity search. The goal is to serve targeted ads to users without revealing their personal information to the ad brokers, such as Google and Facebook. This also implies that the system must hide the ads that the ad brokers are sending out from the ad brokers since knowing the attributes of the displayed ads is also information revealing.

While privacy is the foremost concern of our work, we aim to maintain as much accuracy as possible without compromising run time as well.

2 Related Works

Locality-sensitive hashing (LSH) is the most prominent technique in similarity search in practice. Recently, there has been work on improving LSH techniques in terms of accuracy by utilizing the data [1] as well preserving-privacy LSH in sublinear time [3] [8].

Other works regarding similarity search pointed towards other techniques such as dimensionality reduction. Specifically, the use of locality preserving space-filling lines such as the Hilbert curve can was used to map points in d dimensions into a single dimension. This allowed for the use of b-trees for an approximate nearest-neighbor search [7]. However, their original protocol was not aimed to preserve privacy. Our work, greatly based off of their techniques, adds privacy for both clients and servers.

3 Setup

3.1 Entities

There exists three main entities in a ad retrieval protocol. First is the Ad Broker which is a company that stores ads submitted by advertising companies. Well known examples are Google and Yahoo!. The broker controls the whole ad retrieval system. Brokers often display these ads on their own websites, but they also create deals with other websites to show ads to users. The websites that display ads are known as the Publishers. Well-known examples are New York Times, YouTube, and Wired. Lastly, Users are the targets of the ad brokers that view ads on web browsers.

3.2 Representation of Ads and Users

Users typically have binary profile vectors several thousands of dimensions long that encode various personal information regarding location, hobbies, and demographic. Similarly, the ads contained by Ad Brokers each have a feature vector that encode information regarding the target audience of the ads. Both of these vectors can be thought of as point in d-dimensional space where d is the the size of the dimensions. Similarity search utilize different notions of distance measure such as Jaccard or cosine. The aim is then to find ads with feature vectors closest to the user profile. This can be described as the approximate nearest neighbors problem.

While the original vectors are often thousands of dimensions long, the contents are often extremely sparse meaning that they mainly consist of 0's. In the real world, most people don't have hundreds of hobbies or interests. The sparseness of these vectors allows us to greatly reduce dimensionality using the

Johnson-Lindenstrauss transform (JL) or hashing-based techniques such as MinHash [2]. The remainder of the paper will use d to represent the number of the dimensions of the transformed vectors.

3.3 Similarity Search Algorithms

Currently, common methods for solving the approximate nearest neighbors problem use locality sensitive hashing (LSH). Each point in d-dimensions is mapped to a single number of b-bits. A common way of doing this is using b hyperplanes in d dimensions, each splitting the set of points into two groups. Each hyperplane defines a certain bit, and points that are close together are more likely to have the same hash. This algorithm currently does not preserver privacy, and upon further investigation, requires the use of fully homomoprhic encyrption (FHE) which is highly inefficient in practice.

Despite using a transformation to reduce dimensions, the dimensions of the vectors still makes the nearest-neighbor search difficult. We can use locality-preserving space-filling curves to reduce the number of dimensions to one to be able to apply linear operations. The importance of linear operations is that it allows us to use *additively* homomorphic encryption (AHE) which is much more efficient than FHE.

We use the Hilbert curve (a recursive space filling curve) to accomplish to map d dimensional points to a single line, allowing us to define each ad by its Hilbert distance on the line. This allows us to created a sorted list of ads by Hilbert distance where we essentially find the approximate nearest neighbor using linear regression. This protocol is very similar to [7], replacing the use of B-trees with a privacy-preserving data structure built using only linear regressions. To achieve theoretical guarantees of accuracy, d different sorted lists must be constructed where the data is shifted d times. Proof of this is included in [7]. Unfortunately, having multiple copies of the data is a requirement for correctness, even when using more standard techniques such as LSH [4].

3.4 User Setup

Typically the user profile is stored by the ad broker (server side) as it is constructed by the server from observing user behaior online. In our protocol, we opt for having the profile constructed and stored on the client (e.g. browser), as it can both lead to more accurate targeting and ensures privacy from the broker.

3.5 Ad Broker Setup

The data structure used by the Ad Broker to find the ads is a *Learned Index Structure* [6]. Every point in the sorted list can be thought of as a point (Hilbert distance, index) in a 2D Cartesian grid. These points make a cumulative distribution function (CDF) in which we can use linear regression to approximate

the index of the nearest neighbor by plugging the Hilbert distance of the user profile into the line of best fit. This only requires use of additively homomorphic encryption (AHE) which is much more efficient than FHE. A single linear regression isn't sufficient to achieve high accuracy, so we recursively apply linear regression on smaller subsets of points. The resulting index of each recursive call is used to determine the following subset of ads.

The learned index uses a tree structure where the nodes in every layer represents bins of ads. The root node approximates the entire CDF. Each node has w children, making the size of any layer w^{depth}. Each child represents a smaller interval of ads of its parent, each interval equally-spaced by Hilbert distance. If 0 is the minimum Hilbert distance and Hm is the maximum Hilbert distance, each node in the ith layer represents an an interval of $Hw_i = Hm/w^i$.

The jth node in the ith layer contains the slope (m_{ij}), y-intercept (b_{ij}), and x-intercept (ω_{ij}).

This result of the learned index gives an index for an approximate nearest neighbor. This would be done for each of the d lists in parallel as described in [7].

4 Protocol

In order to achieve theoretical guarantees of accuracy, the following protocol must be completed $(d+1)$ times. In each iteration, the original JL transformed vectors for ads and the user profile are shifted as described in.

4.1 Interactive Index Retrieval

1. The User computes Q, the Hilbert distance for the JL transformed profile vector.

2. The User only needs to compute the result of a single node for the approximate nearest neighbor in a layer. However this would reveal which nodes are being used to the Ad Broker, revealing information about Q and the user profile. Thus, all the nodes in the tree must be used. For each layer i of the learned index:

   (a) In any layer, there is an index $ind_i \in [1, w^i]$ that represents the node in the layer of the learned index that is being used for approximation. For the first layer, ind_0 is always 1 as there is only 1 node. The User typically doesn't know ind_i but rather $ind'_i = ind_i + r_i \ mod \ w^i$, where r_i is a random positive value known to the Ad Broker and w^i is the size of the layer.

   (b) The User computes a vector g and h where each have w^i elements. Every value of g is 1 except for a 0 at index ind'_i, and every value of h is 0 except for Q at ind'_i. All values of 0's and 1's must also be encrypted to hide ind'_i.

(c) The Ad Broker receives g and h, and rotates the vectors by r_i, moving the 1 in g and Q in h to ind. For every j, the server transforms g by multiplying each g_j with ω_{ij}. g and h are combined into s, where $s_j = g_j + h_j$. The resulting array s has ω_{ij} at every location except for Q at ind. s is then transformed by $s_j = m_{ij} \cdot s_j + b_{ij}$. Note that $m_{ij} \cdot \omega_{ij} + b_{ij} = 0$, thus s contains 0 at every location except for $m_{i(ind)} \cdot Q + b_{i(ind)}$ at ind, which is the approximate index for nearest neighbor. The Ad Broker computes p, the sum of the values of s. p' is then calculated as $p' = p + r_{i+1} \cdot Hw_{i+1}$.

(d) The User receives m' and decrypts it. The user than computes $ind'_{i+1} = m'/Hw_{i+1}\ mod\ w^i$ for the next layer.

3. The result of the final m' is used for ad retrieval.

4.2 Ad Retrieval

The User now has an approximate index m' for the nearest neighbors to query using PIR. The Ad Broker will compute $m = m' - r_{i+1} \cdot Hw_{i+1}$ during the PIR protocol to return the ad at the proper index. In order to retrieve the approximate k-nearest neighbors, as multiple ads are typically displayed, the User queries a sub-sequence of ads in the sorted list. The User receives the $k/2$ ads to the left and right of m along with the ad at m.

5 Analysis

Theorem 1 (Correctness)**.** *The protocol of Section 4 is correct and outputs the approximate nearest neighbor to the query.*

Proof. Correctness follows from the theoretical guarantees of the ANN algorithm described in [7]. By close inspection of our protocol, we evaluate a learned index structure in a privacy preserving way which emulates the binary tree used in [7]. □

Theorem 2 (Client Privacy)**.** *By executing the protocol in 4.1, the server does not learn anything of the client's profile vector including its Hilbert value.*

Proof. In each interaction, the server only sees two encrypted vectors, hiding the index of the node used in the layer as well as the Hilbert value of the client's profile vector. Every node in the learned index is used, so the server can't assume anything of the client. Thus, the server can only learn anything of the client's profile vector if it manages to break the encryption scheme. □

Theorem 3 (Server Privacy)**.** *Theorem 3 (Server Privacy): The client learns nothing of the server's learned index model.*

Proof. The client receives m_i in every interaction with the server which reveals nothing of the server's learned index model since m_i is shifted by an unknown value $r_i \cdot Hw_i$. □

5.1 Accuracy & Runtime

Ad Broker's keep the feature vectors of ads and user profiles secret, as their exact similarity search algorithms are secret and important aspect of their revenue. Thus, it is difficult to test the protocol on real world data, but we used a machine learning data-set [5].

Due to the difference between LSH and our protocol, it was challenging to accurately compare accuracy. We opted for whether or not the nearest neighbor was found as a *pass* or *fail*. For our protocol, a parameter was used to allow for some error in the exact nearest neighbor search some k away from the determined index. This was done for all $(d+1)$ learned indexes to check if the nearest neighbor was found in any of them. LSH was tested using a constant of 10 bits in its hash representation. The resulting accuracy for both methods were almost the same up to 50 dimensions, both ending up around 92%. This however, generated an unreasonably large amount of run-time for our protocol as the creation for the learned index and querying started taking several seconds long.

In practice, it is un-acceptable for ad selection to really take more than a second, and it makes more more sense to only shift the data a few times instead of d times, trading off some accuracy for runtime. Using a static value of 10 shifts for our protocol, LSH generally performed better. In Figure 1 we show that at 50 dimensions LSH had around a 92% while our protocol had around 89% accuracy.

Testing on randomized data points, our protocol tended to yield low accuracy. However, the feature vectors for ads are most likely clustered as similar ads have similar feature vectors, and the data is generally sparse. Thus, even without being able to test using real feature vectors and user profiles, we believe additional privacy guarantees is worthwhile trade-off for any anticipated accuracy decrease.

6 Conclusion

We present a privacy preserving protocol for the approximate nearest neighbor problem. We adapt an existing algorithm for finding approximate nearest neighbors to a client-server setting where a client holds a secret profile vector and the server holds a database of points. Our protocol allows the client to find the indices of points in the database that most closely match the query, without leaking the query to the server and without revealing any additional information to the client (the server's database of points remains secret to the client).

Our protocol can be used to match user profiles with targeted advertisements in a privacy-preserving way, which we hope can pave the way for a more privacy-conscious online advertising eco-system.

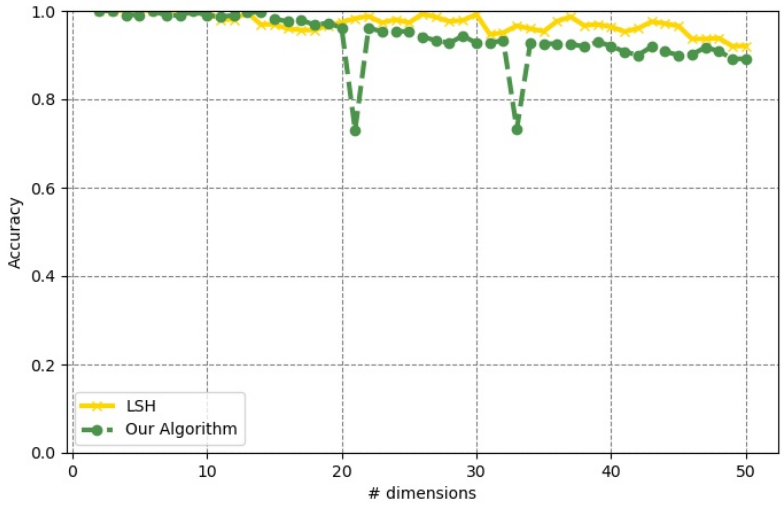

Figure 1: Comparison of our proposed protocol with LSH

7 Acknowledgements

I thank the MIT PRIMES program for giving me the opportunity to work on this project. I would also like to thank my mentors Sacha Servan-Schreiber and Kyle Hogan from the MIT CS department.

References

[1] A. Andoni and I. Razenshteyn. Optimal data-dependent hashing for approximate near neighbors, 2015.

[2] A. Z. Broder. On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*, pages 21–29. IEEE, 1997.

[3] H. Chen, I. Chillotti, Y. Dong, O. Poburinnaya, I. Razenshteyn, and M. S. Riazi. Sanns: Scaling up secure approximate k-nearest neighbors search, 2019.

[4] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262, 2004.

[5] D. Dua and C. Graff. UCI machine learning repository, 2017.

[6] T. Kraska, A. Beutel, E. H. Chi, J. Dean, and N. Polyzotis. The case for learned index structures, 2017.

[7] S. Liao, M. A. Lopez, and S. T. Leutenegger. High dimensional similarity search with space filling curves. In *Proceedings 17th International Conference on Data Engineering*, 2001.

[8] M. S. Riazi, B. Chen, A. Shrivastava, D. S. Wallach, and F. Koushanfar. Sub-linear privacy-preserving near-neighbor search. *IACR Cryptol. ePrint Arch.*, 2019:1222, 2019.