© 2018 IJSRSET | Volume 4 | Issue 9 | Print ISSN: 2395-1990 | Online ISSN : 2394-4099
Themed Section : Engineering and Technology

Privacy Preserving Collaborative Model Document Clustering Using TF-IDF Approach

T. G. Babu*[1], E. Anitha[2]

[1]Assistant Professor, PG &Research Department of Computer Science and Science and Applications, Arignar Anna Govt Arts College Arcot Road, Cheyyar, Vellore, Tamil Nadu, India

[2] M.Phil (CS) Research Scholar PG &Research Department of Computer Science and Science and Applications, Arignar Anna Govt Arts College Arcot Road, Cheyyar, Vellore, Tamil Nadu, India

ABSTRACT

With the expanded popularity of public computing infrastructures (e.g., cloud platform), it has been more advantageous than any other time in recent days for distributed users (across the Internet) to perform collaborative learning through the shared infrastructure. While the potential advantages of (collective) machine learning can be gigantic, and the large-scale training data may posture generous privacy risks. In other words, centralized collection of data from different participants may raise great concerns in data confidentiality and privacy. For instance, in certain application scenarios such as healthcare, individuals/patients may not reveal their sensitive information (e.g., protected health data) to any other person, and the exposure of such exclusive information is prohibited by the laws or controls of HIPAA1. To manage such privacy issues, a clear approach is to encode sensitive information before sharing it. However, data encryption hinders data utilization and computation, making it hard to proficiently perform (community) machine learning compared with the case in plaintext domain.

Keywords: Continuous Bag of words, Skip-gram model, Google DeepMind, AlphaGo, Healthcare provider

I. INTRODUCTION

A. General Study

Overcome one of the mainstays of information technology. VER the past two decades, machine learning has be With the advancement in neural-network-based learning methods, major breakthroughs have been made for classic artificial intelligence tasks, such as speech/image/text recognition, automatic translation, and web page ranking. This is enabled, in part, by the availability of a huge volume of high-quality data used for training neural networks. For example, Google DeepMind developed a program named AlphaGo, which trains on about 30 million positions from expert games to play the board game Go, and has beaten professional human players without handicaps. Compared to training with only local dataset, collaborative learning can improve the accuracy of resulting models by incorporating more representative data.

B. Goal

Our goal is to ensure that the remote server S and the crypto service provider HEALTH CARE PROVIDER cannot learn anything about the users' data beyond what is revealed by the results of the learning algorithm.

C. Problem Definitions

Consider the problem of computing continuous vector representations of words over encrypted data files collected by a central remote server. At a high

level, we target at a system composed of three major parties: multiple participating users (i.e., PHR Owners), a central remote server S and a crypto service provider HEALTH CARE PROVIDER. More specifically, there are n users in total, denoted as ui (i = 1 : : : n), each of which owns a private data file fi and wants to perform collaborative neural network learning with all other participating users. In other words, they will contribute their private data files in the encrypted form to the central remote server. After receiving the encrypted data, the remote server S performs training over the contributed data and produces high-quality word vectors along with a model, which can be used later for various natural language processing (NLP) tasks.

Collusion between them is highly impossible as it will damage their reputation. We further assume that the remote server S and the HEALTH CARE PROVIDER are both honest-but-curious entities meaning that they will run the protocol exactly as specified without any deviations, but try to learn extra information from their views of the protocol.

D. Challenges

To addresses three major challenges induced by data encryption.

- ✓ First, to overcome the difficulty that the plaintext space of the cryptosystem adopted in our construction only constituted integral numbers, we introduce fixed-point data type to deal with real numbers involved in computations.
- ✓ Second, to facilitate the efficient evaluation of non-linear polynomials (e.g., multiplication and exponentiation operation) in the encrypted form, we leverage a secure multiplication protocol combined with the packing technique to be deployed on two parties.
- ✓ Third, to enable the private computation of activation function (also transcendental function) used in neural networks, we make an approximation by leveraging customized series expansion which incurs only a little loss in accuracy.

- ✓ **Contributions**
- ✓ Our main contributions are summarized as follows
- ✓ To the best of our knowledge, this paper is the first to provide privacy preservation for neural network learning algorithms to train distributed word vectors (which are of great significance to numerous NLP applications) over large-scale encrypted data from multiple participants.
- ✓ We first leverage a couple of arithmetic primitives (e.g., multiplication and fixed-point representation) on encrypted data, and then design a new technique which enables the secure computation of activation function. These arithmetics over ciphertexts are served as components of our tailored construction.
- ✓ We present a thorough analysis regarding to privacy and efficiency of our proposed construction. To further demonstrate its practicality and effectiveness, we conduct experimental evaluations on representative real-world datasets and make comparison with results obtained in the plaintext domain.

II. LITERATURE SURVEY

A. Mastering The Game Of Go With Deep Neural Networks And Tree Search

In that paper, **D. Silver & team** says that a long-standing goal of artificial intelligence is an algorithm that learns, tabula rasa, superhuman proficiency in challenging domains. Recently, AlphaGo became the first program to defeat a world champion in the game of Go. The tree search in AlphaGo evaluated positions and selected moves using deep neural networks. These neural networks were trained by supervised learning from human expert moves, and by reinforcement learning from selfplay. Here, we introduce an algorithm based solely on reinforcement learning, without human data, guidance, or domain knowledge beyond game rules. AlphaGo becomes its

own teacher: a neural network is trained to predict AlphaGo's own move selections and also the winner of AlphaGo's games. This neural network improves the strength of tree search, resulting in higher quality move selection and stronger self-play in the next iteration. Starting tabula rasa, our new program AlphaGo Zero achieved superhuman performance, winning 100-0 against the previously published, champion-defeating AlphaGo.

Drawbacks

1) These systems have outperformed humans in computer games such as Atari 6, 7 and 3D virtual environments.
2) However, the most challenging domains in terms of human intellect – such as the game of Go, widely viewed as a grand challenge for artificial intelligence require precise and sophisticated look ahead in vast search spaces.
3) Fully general methods have not previously achieved human-level performance in these domains.

B. On Data Banks And Privacy Homomorphisms

R. L. Rivest & team defines Encryption as a well—known technique for preserving the privacy of sensitive information. One of the basic, apparently inherent, limitations of this technique is that an information system working with encrypted data can at most store or retrieve the data for the user; any more complicated operations seem to require that the data be decrypted before being operated on. This limitation follows from the choice of encryption functions used, however, and although there are some truly inherent limitations on what can be accomplished, we shall see that it appears likely that there exist encryption functions which permit encrypted data to be operated on without preliminary decryption of the operands, for many sets of interesting operations. These special encryption functions we call "privacy homeomorphisms"; they form an interesting subset of arbitrary encryption schemes (called "privacy transformations")

Drawbacks

1) The systems programmers would presumably have access to the sensitive information.
2) The loan company therefore decides to encrypt all of its data kept in the data bank and to maintain a policy of only decrypting data at the home office data will never be decrypted by the time shared computer.
3) The situation is where the wavy line encircles the physically secure premises of the loan company.

C. Privacy-Preserving Ridge Regression On Hundreds Of Millions Of Records

V. Nikolaenko explains that Linear regression with 2-norm regularization (i.e., ridge regression) is an important statistical technique that models the relationship between some explanatory values and an outcome value using a linear function. In many applications (e.g., predictive modeling in personalized health-care), these values represent sensitive data owned by several different parties who are unwilling to share them. In this setting, training a linear regression model becomes challenging and needs specific cryptographic solutions. This problem was elegantly addressed by Nikolaenko et al. in S&P (Oakland) 2013. They suggested a two-server system that uses linearly-homomorphic encryption (LHE) and Yao's two-party protocol (garbled circuits). In this work, we propose a novel system that can train a ridge linear regression model using only LHE (i.e., without using Yao's protocol). This greatly improves the overall performance (both in computation and communication) as Yao's protocol was the main bottleneck in the previous solution. The efficiency of the proposed system is validated both on synthetically-generated and real-world datasets.

Drawbacks

1) We would like to use a given linear regression method in order to predict the weight of a baby at birth on the basis of some ultrasound measurements made during the last month of

pregnancy (e.g., head circumference, femur length, . . .).

2) On one hand, in order to avoid computing a biased model, we would like to run the selected learning algorithm on data points collected in different hospitals in various locations.
3) On the other hand, each hospital legally cannot share (in the clear) patients' sensitive data (the measurements) with other hospitals or with a third party (e.g., a cloud-computing server).
4) This real-life case exemplifies the challenge on which we focus on: training a linear regression model on joint data that must be kept confidential and/or are owned by multiple parties.
5) Moreover, we want to run such collaborative analysis without exposing an entity's sensitive data to any other party in the system (i.e., no entity in the system is trusted to handle the data in the clear).

D. Privacy-Preserving Matrix Factorization

V. Nikolaenko and team state that Recommender systems typically require users to reveal their ratings to a recommender service, which subsequently uses them to provide relevant recommendations. Revealing ratings has been shown to make users susceptible to a broad set of inference attacks, allowing the recommender to learn private user attributes, such as gender, age, etc. In this work, we show that a recommender can profile items without ever learning the ratings users provide, or even which items they have rated. We show this by designing a system that performs matrix factorization, a popular method used in a variety of modern recommendation systems, through a cryptographic technique known as garbled circuits. Our design uses oblivious sorting networks in a novel way to leverage sparsity in the data. This yields an efficient implementation, whose running time is Θ(M log2 M) in the number of ratings M. Crucially, our design is also highly parallelizable, giving a linear speedup with the number of available processors. We further fully implement our system, and demonstrate that even on commodity hardware with 16 cores, our privacy-preserving implementation can factorize a matrix with 10K ratings within a few hours.

Drawbacks

1) Our security guarantees will hold under the honest but curious threat model.
2) In other words, the RecSys and HEALTH CARE PROVIDER follow the protocols we propose as prescribed.
3) However, these interested parties may elect to analyze protocol transcripts, even off-line, in order to infer some additional information.
4) We further assume that the recommender and HEALTH CARE PROVIDER do not collude

E. Efficient Privacy-Preserving Matrix Factorization Via Fully Homomorphic Encryption

S. Kim and team said that Recommendation systems become popular in our daily life. It is well known that the more the release of users' personal data, the better the quality of recommendation. However, such services raise serious privacy concerns for users. In this paper, focusing on matrix factorization-based recommendation systems, we propose the first privacy-preserving matrix factorization using fully homomorphic encryption. On inputs of encrypted users' ratings, our protocol performs matrix factorization over the encrypted data and returns encrypted outputs so that the recommendation system knows nothing on rating values and resulting user/item profiles. It provides a way to obfuscate the number and list of items a user rated without harming the accuracy of recommendation, and additionally protects recommender's tuning parameters for business benefit and allows the recommender to optimize the parameters for quality of service. To overcome performance degradation caused by the use of fully homomorphic encryption, we introduce a novel data structure to perform computations over encrypted vectors, which are essential operations for matrix factorization, through secure 2-party computation in part. With the data structure, the

proposed protocol requires dozens of times less computation cost over those of previous works. Our experiments on a personal computer with 3.4 GHz 6-cores 64 GB RAM show that the proposed protocol runs in 1.5 minutes per iteration. It is more efficient than Nikolaenko et al.'s work proposed in CCS 2013, in which it took about 170 minutes on two servers with 1.9 GHz 16-cores 128 GB RAM.

Drawbacks

1) A user (client) with some private data (e.g., ratings) would like to buy a recommendation service to efficiently figure out the most favorable products from a large number of potential candidates.
2) A service provider (e.g., Amazon) has already collected a large database of ratings given by its users on sold items and wishes to monetize its data by selling Recommendation as a Service (RaaS).
3) Different from existing recommender systems, in our scenario the client is unwilling to expose her data to the service provider due to the worries of privacy leakage.
4) At the same time, commercial concerns or requirements may prevent the service provider from releasing its trained recommendation model to the public.
5) In addition, releasing a trained model may also bring privacy risks to the users in the service provider's database.

III. RELATED WORK

A. Embeddings

In very simplistic terms, Word Embeddings are the texts converted into numbers and there may be different numerical representations of the same text.
Machine Learning algorithms and almost all Deep Learning Architectures are incapable of processing *strings* or *plain text* in their raw form. They require numbers as inputs to perform any sort of job, be it classification, regression etc. in broad terms. And with the huge amount of data that is present in the text format, it is imperative to extract knowledge out of it and build applications. Some real world applications of text applications are – sentiment analysis of reviews by Amazon etc., document or news classification or clustering by Google etc.

Let us now define Word Embeddings formally. A Word Embedding format generally tries to map a word using a dictionary to a vector. Let us break this sentence down into finer details to have a clear view.
Take a look at this example – **sentence**=" Word Embeddings are Word converted into numbers"

A *word* in this **sentence** may be "Embeddings" or "numbers" etc.
A *dictionary* may be the list of all unique words in the **sentence.** So, a dictionary may look like – ['Word','Embeddings','are','Converted','into','numbers']

B. Prediction Based Vector

Word2vec is not a single algorithm but a combination of two techniques – CBOW (Continuous bag of words) and Skip-gram model. Both of these are shallow neural networks which map word(s) to the target variable which is also a word(s). Both of these techniques learn weights which act as word vector representations. Let us discuss both these methods separately and gain intuition into their working.

a) **CBOW (Continuous Bag of words)**

The way CBOW work is that it tends to predict the probability of a word given a context. A context may be a single word or a group of words. But for simplicity, I will take a single context word and try to predict a single target word.

Suppose, we have a corpus C = "Hey, this is sample corpus using only one context word." and we have defined a context window of 1. This corpus may be converted into a training set for a CBOW model as follow. The input is shown below. The matrix on the

right in the below image contains the one-hot encoded from of the input on the left.

Input	Output		Hey	This	is	sample	corpus	using	only	one	context	word
Hey	this	Datapoint 1	1	0	0	0	0	0	0	0	0	0
this	hey	Datapoint 2	0	1	0	0	0	0	0	0	0	0
is	this	Datapoint 3	0	0	1	0	0	0	0	0	0	0
is	sample	Datapoint 4	0	0	1	0	0	0	0	0	0	0
sample	is	Datapoint 5	0	0	0	1	0	0	0	0	0	0
sample	corpus	Datapoint 6	0	0	0	1	0	0	0	0	0	0
corpus	sample	Datapoint 7	0	0	0	0	1	0	0	0	0	0
corpus	using	Datapoint 8	0	0	0	0	1	0	0	0	0	0
using	corpus	Datapoint 9	0	0	0	0	0	1	0	0	0	0
using	only	Datapoint 10	0	0	0	0	0	1	0	0	0	0
only	using	Datapoint 11	0	0	0	0	0	0	1	0	0	0
only	one	Datapoint 12	0	0	0	0	0	0	1	0	0	0
one	only	Datapoint 13	0	0	0	0	0	0	0	1	0	0
one	context	Datapoint 14	0	0	0	0	0	0	0	1	0	0
context	one	Datapoint 15	0	0	0	0	0	0	0	0	1	0
context	word	Datapoint 16	0	0	0	0	0	0	0	0	1	0
word	context	Datapoint 17	0	0	0	0	0	0	0	0	0	1

Figure 1. Matrix

This matrix shown in the above image is sent into a shallow neural network with three layers: an input layer, a hidden layer and an output layer. The output layer is a softmax layer which is used to sum the probabilities obtained in the output layer to 1. Now let us see how the forward propagation will work to calculate the hidden layer activation.

Let us first see a diagrammatic representation of the CBOW model.

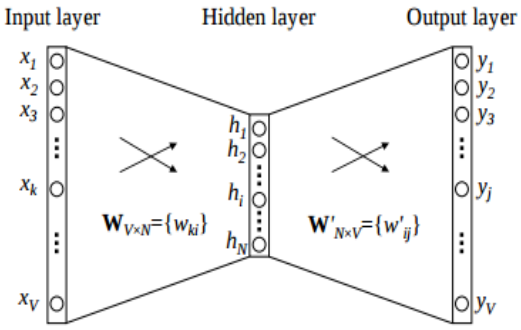

Figure 2. CBOW Model

The flow is as follows:

1) The input layer and the target, both are one-hot encoded of size [1 X V]. Here V=10 in the above example.
2) There are two sets of weights. One is between the input and the hidden layer and second between hidden and output layer. Input-Hidden layer matrix size =[V X N] , hidden-Output layer matrix size =[N X V] : Where N is the number of dimensions we choose to represent our word in. It is arbitary and a hyper-parameter for a Neural Network. Also, N is the number of neurons in the hidden layer. Here, N=4.
3) There is a no activation function between any layers.(More specifically, I am referring to linear activation)
4) The input is multiplied by the input-hidden weights and called hidden activation. It is simply the corresponding row in the input-hidden matrix copied.
5) The hidden input gets multiplied by hidden-output weights and output is calculated.
6) Error between output and target is calculated and propagated back to re-adjust the weights.
7) The weight between the hidden layer and the output layer is taken as the word vector representation of the word.
8) We saw the above steps for a single context word. Now, what about if we have multiple context words? The image below describes the architecture for multiple context words.

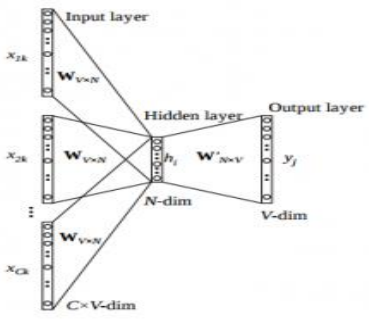

Figure 3. CBOW Architechture

Below is a matrix representation of the above architecture for an easy understanding.

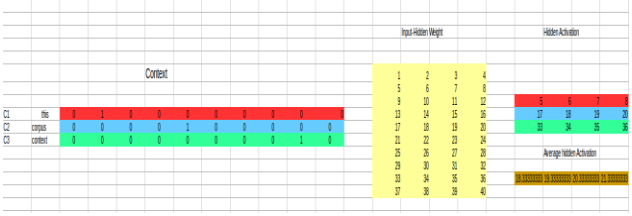

Figure 4. Matrix representation

The image Figure above takes 3 context words and predicts the probability of a target word. The input can be assumed as taking three one-hot encoded vectors in the input layer as shown above in red, blue and green.

So, the input layer will have 3 [1 X V] Vectors in the input as shown above and 1 [1 X V] in the output layer. Rest of the architecture is same as for a 1-context CBOW.

The steps remain the same, only the calculation of hidden activation changes. Instead of just copying the corresponding rows of the input-hidden weight matrix to the hidden layer, an average is taken over all the corresponding rows of the matrix. We can understand this with the above figure. The average vector calculated becomes the hidden activation. So, if we have three context words for a single target word, we will have three initial hidden activations which are then averaged element-wise to obtain the final activation.

The differences between MLP and CBOW are mentioned below for clarification:
1) The objective function in MLP is a MSE(mean square error) whereas in CBOW it is negative log likelihood of a word given a set of context i.e -log(p(wo/wi)), where p(wo/wi) is given as

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^{\top} v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^{\top} v_{w_I}\right)}$$

wo : output word wi: context words
2) The gradient of error with respect to hidden-output weights and input-hidden weights are different since MLP has sigmoid activations (generally) but CBOW has linear activations. The method however to calculate the gradient is same as an MLP.

Advantages of CBOW:
1) Being probabilistic is nature, it is supposed to perform superior to deterministic methods (generally).
2) It is low on memory. It does not need to have huge RAM requirements like that of co-occurrence matrix where it needs to store three huge matrices.

Disadvantages of CBOW:
1) CBOW takes the average of the context of a word (as seen above in calculation of hidden activation). For example, Apple can be both a fruit and a company but CBOW takes an average of both the contexts and places it in between a cluster for fruits and companies.
2) Training a CBOW from scratch can take forever if not properly optimized.

b) Skip – Gram Model

Skip – gram follows the same topology as of CBOW. It just flips CBOW's architecture on its head. The aim of skip-gram is to predict the context given a word. Let us take the same corpus that we built our CBOW model on. C="Hey, this is sample corpus using only one context word." Let us construct the training data.

Input	Output(Context1)	Output(Context2)
Hey	this	<padding>
this	Hey	is
is	this	sample
sample	is	corpus
corpus	sample	corpus
using	corpus	only
only	using	one
one	only	context
context	one	word
word	context	<padding>

Figure 5. Training Data

The input vector for skip-gram is going to be similar to a 1-context CBOW model. Also, the calculations up to hidden layer activations are going to be the same. The difference will be in the target variable. Since we have defined a context window of 1 on both the sides, there will be "**two" one hot encoded target variables** and "**two" corresponding outputs** as can be seen by the blue section in the image.

Two separate errors are calculated with respect to the two target variables and the two error vectors obtained are added element-wise to obtain a final

error vector which is propagated back to update the weights.

The weights between the input and the hidden layer are taken as the word vector representation after training. The loss function or the objective is of the same type as of the CBOW model.

The skip-gram architecture is shown below.

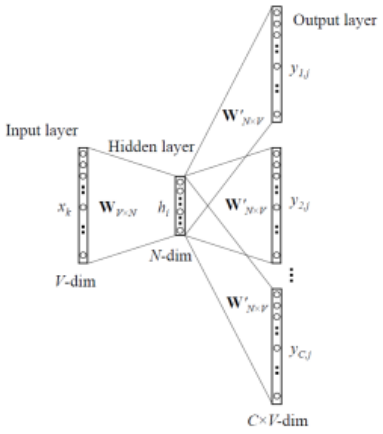

Figure 6. Skip-gram architecture

For a better understanding, matrix style structure with calculation has been shown below.

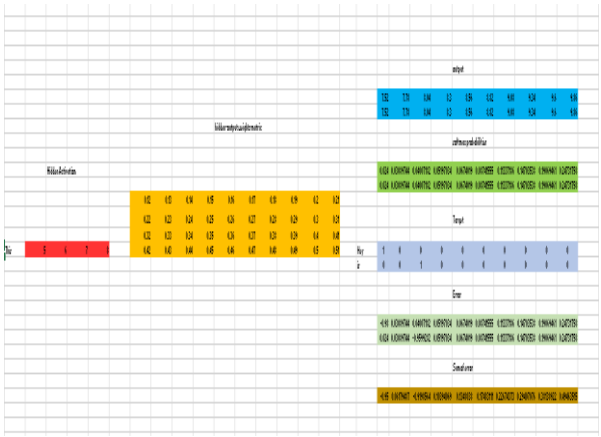

Figure 7. Matrix representation

Let us break down the above image.
Input layer size – [1 X V], Input hidden weight matrix size – [V X N], Number of neurons in hidden layer – N, Hidden-Output weight matrix size – [N X V], Output layer size – C [1 X V]

In the above example, C is the number of context words=2, V= 10, N=4

1) The row in red is the hidden activation corresponding to the input one-hot encoded vector. It is basically the corresponding row of input-hidden matrix copied.
2) The yellow matrix is the weight between the hidden layer and the output layer.
3) The blue matrix is obtained by the matrix multiplication of hidden activation and the hidden output weights. There will be two rows calculated for two target(context) words.
4) Each row of the blue matrix is converted into its softmax probabilities individually as shown in the green box.
5) The grey matrix contains the one hot encoded vectors of the two context words(target).
6) Error is calculated by substracting the first row of the grey matrix(target) from the first row of the green matrix(output) element-wise. This is repeated for the next row. Therefore, for n target context words, we will have n error vectors.
7) Element-wise sum is taken over all the error vectors to obtain a final error vector.
8) This error vector is propagated back to update the weights.

Advantages of Skip-Gram Model

1) Skip-gram model can capture two semantics for a single word. i.e it will have two vector representations of Apple. One for the company and other for the fruit.
2) Skip-gram with negative sub-sampling outperforms every other method generally.

This is an excellent interactive tool to visualize CBOW and skip gram in action. I would suggest you to really go through this link for a better understanding.

IV. ALGORITHM AND ARCHITECTURE

A. Architecture

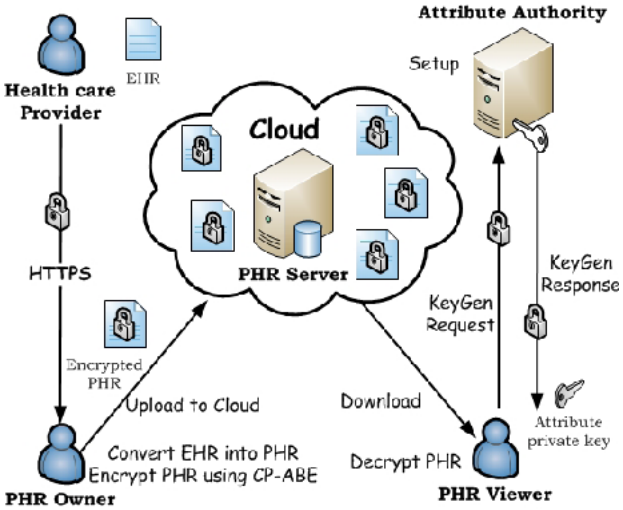

Figure 8. Architecture Diagram

This system has the following two-phase architecture:

Phase 1 (merging the local datasets)

HEALTH CARE PROVIDER generates the key pair „*sk*; *pk*", stores *sk* and makes *pk* public; each DO*i* sends to MLE specific ciphertexts computed using *pk* and the values in D*i*. MLE uses the ciphertexts received and the homomorphic property of the underling encryption scheme in order to obtain encryptions of *A* and ***b*** (coefficient matrix and vector in (1)).

Phase 2 (computing the model)

MLE uses the ciphertexts Enc*pk*„*A*" and Enc*pk*„***b***" and private random values in order to obtain encryptions of new values that we call "masked data"; these encryptions are sent to the HEALTH CARE PROVIDER; the latter decrypts and runs a given algorithm on the masked data. The output of this computation ("masked model") is a vector $\tilde{\boldsymbol{w}}$ that is sent back from the HEALTH CARE PROVIDER to the MLE. The latter computes the output $\boldsymbol{w}^*$ from $\tilde{\boldsymbol{w}}$.

Having described our approach, we now sketch a system architecture that might be used to implement it. This is divided into two parts:

a. Residing in the cloud, the first part is responsible for constructing a *shared* model using batch learning; and
b. Residing on each individual user's device, the second part tunes the model from the first part using the locally available data, resulting in a *personal* model.

We identify five components in this architecture:

1) The **batch training module** resides in the cloud, and is responsible for training a *shared* model as the starting point using public, or private but shared, datasets that it also maintains. As this component may need to support multiple applications, it will provide a collection of different machine learning algorithms to build various needed models. It may also need to perform more traditional, large scale processing, but can easily be built using modern data processing frameworks designed for datacenters such as Mllib or GraphLab.

2) The **distribution module** resides on users' devices and is responsible for obtaining the *shared* model and maintaining it locally. In the case of very large scale deployments, standard content distribution or even peerto-peer techniques could be used to alleviate load on the cloud service.

3) The **personalization module** builds a *personal model* by refining the model parameters of the shared model using the personal data available on the user's device. This module will also require a repository of different learning algorithms, but the nature of personal computational devices means that there will be greater resource constraints applied to the performance and efficiency of the algorithm implementation.

4) The **communication module** handles all the communications between peers or those between an individual node and the server. Nodes can register themselves with the server, on top of which we can

implement more sophisticated membership management.

5) The **inference module** provides a service at the client to respond to model queries, using the most refined model available. In our implementation, we rely on several existing software libraries to provide the more mundane of these functions, e.g., ZeroMQ satisfies most of the requirements of the communication and model distribution modules, and so we do not discuss these further here. There are many toolkits], that provide a rich set of machine learning algorithms for use in the batch training and personalization modules. However, in the case of the latter, we must balance convenience with performance considerations due to the resource-constrained nature of these devices. In light of this, we use a more recent library, Owl, to generate more compact and efficient native code on a range of platforms, and the source code can be obtained from its Github repository.

B. Algorithm

Frequency based Embedding

There are generally three types of vectors that we encounter under this category.

1. Count Vector
2. TF-IDF Vector
3. Co-Occurrence Vector

Let us look into each of these vectorization methods in detail.

1) Count Vector

Consider a Corpus C of D documents {d1,d2…..dD} and N unique tokens extracted out of the corpus C. The N tokens will form our dictionary and the size of the Count Vector matrix M will be given by D X N. Each row in the matrix M contains the frequency of tokens in document D(i).

2) TF-IDF vectorization

This is another method which is based on the frequency method but it is different to the count vectorization in the sense that it takes into account not just the occurrence of a word in a single document but in the entire corpus.

Common words like 'is', 'the', 'a' etc. tend to appear quite frequently in comparison to the words which are important to a document. For example, a document A on Lionel Messi is going to contain more occurrences of the word "Messi" in comparison to other documents. But common words like "the" etc. are also going to be present in higher frequency in almost every document.

3) Co-Occurrence Matrix With A Fixed Context Window

The big idea – similar words tend to occur together and will have similar context for example – apple is a fruit. Mango is a fruit. Apple and mango tend to have a similar context i.e. Fruit.

Before I dive into the details of how a co-occurrence matrix is constructed, there are two concepts that need to be clarified – Co-Occurrence and Context Window.

Co-occurrence – For a given corpus, the co-occurrence of a pair of words say w1 and w2 is the number of times they have appeared together in a Context Window.

Context Window – Context window is specified by a number and the direction.

IMPLEMENTATION MODULES & SCREENSHOTS

A. Modules

1) PHR user Module

This module includes the user registration login details. This module is used to help the client to search the file using the multiple key words concept and get the accurate result list based on the user query. The user is going to select the required file and register the user details and get activation code in mail from the "customerservice404" email before enter the activation code.

2) File Upload Module

This module helps the owner to upload his file with encryption using RSA algorithm. This ensures the files to be protected from unauthorized user.

3) Rank Search Module

These modules ensure the user to search the file that is searched frequently using rank search.

4) File Download Module

This module allows the user to download the file using his secret key to decrypt the downloaded data.

5) View Uploaded and Downloaded File

This module allows the Owner to view the uploaded files and downloaded files

B. Result

1) **Screenshots**

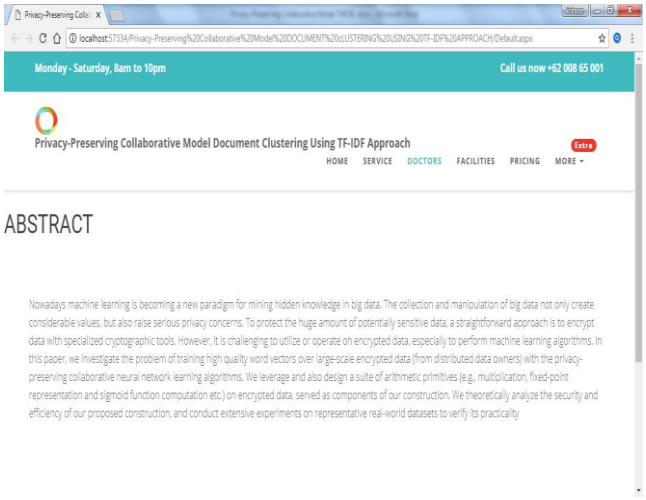

Figure 9. Home Page

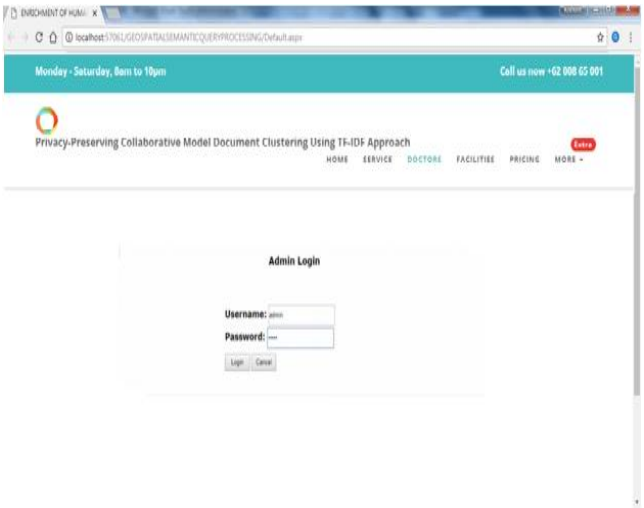

Figure 10. Admin Login

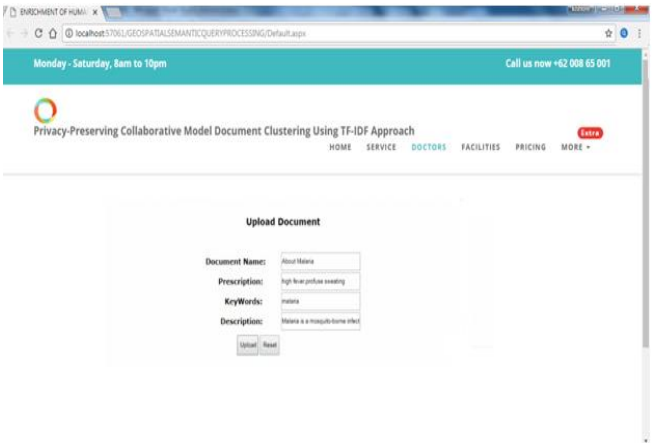

Figure 11. Upload Document

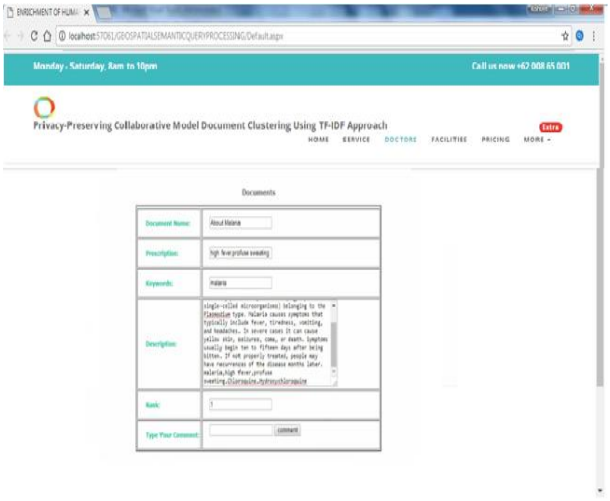

Figure 12. View Document

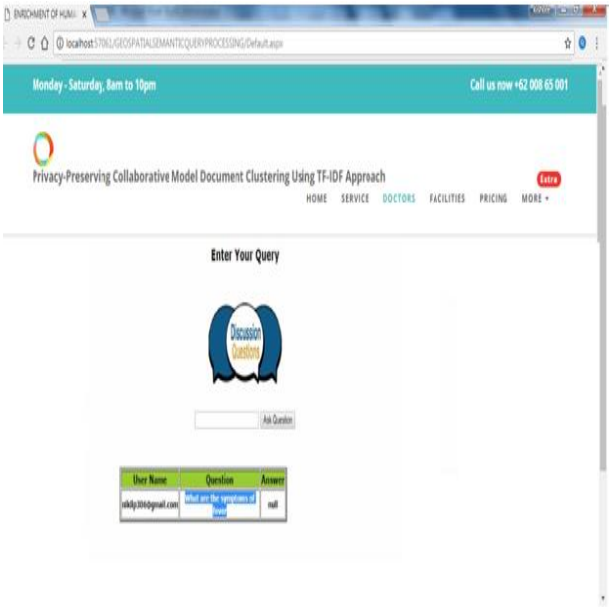

Figure 13. User Search Query

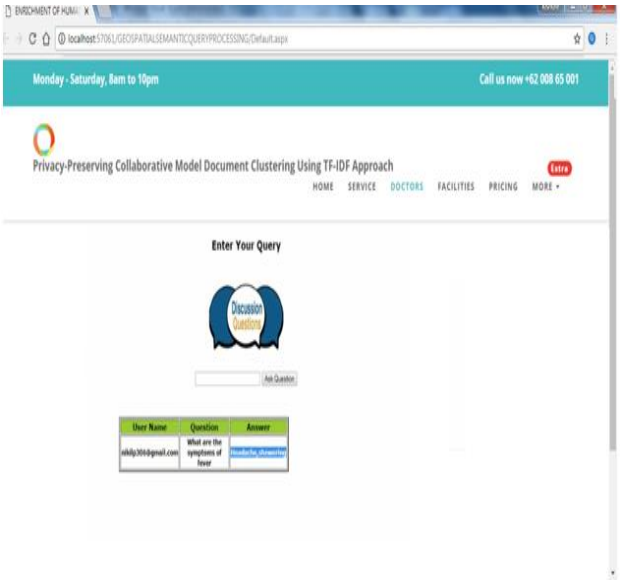

Figure 14. User Search Result

V. CONCLUSION

In this Research Work, we proposed four new security protecting neural system learning plans for preparing word vectors over substantial scale encoded information from different members. In our development, we deliberately utilized a proficient additively homomorphic encryption, and presented a suite of arithmetic primitives over encrypted data to serve as components. Finally, theoretical analyses and extensive experimental evaluations on real-world datasets were conducted to show that our schemes are secure, efficient and practical for use in real-world NLP applications

We trust our work to be significant from the accompanying two points of view. To start with, it gives a novel outline for learning word vectors over encrypted information, which ensures security of the (possibly) sensitive training data. Our methodology may be also applied to other research directions of machine learning, such as convolution neural networks (CNN). Second, it provides multiple cryptographic primitives, which are building blocks for more complex arithmetic operations (*e.g.*, arbitrary non-linear functions).

VI. REFERENCES

[1]. D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot et al., "Mastering the game of go with deep neural networks and tree search," Nature, vol. 529, no. 7587, pp. 484–489, 2016.

[2]. R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," Foundations of secure computation, vol. 4, no. 11, pp. 169–180, 1978.

[3]. V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, "Privacy-preserving ridge regression on hundreds of millions of records," in Proc. of S&P'13. IEEE, 2013, pp. 334–348.

[4]. A. C. Yao, "Protocols for secure computations," in Proc. of FOCS'82. IEEE, 1982, pp. 160–164.

[5]. V. Nikolaenko, S. Ioannidis, U. Weinsberg, M. Joye, N. Taft, and D. Boneh, "Privacy-preserving matrix factorization," in Proc. Of CCS'13. ACM, 2013, pp. 801–812.

[6]. S. Kim, J. Kim, D. Koo, Y. Kim, H. Yoon, and J. Shin, "Efficient privacy-preserving matrix factorization via fully homomorphic encryption," in Proc. of AsiaCCS'16. ACM, 2016, pp. 617–628.

[7]. R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," in Proc. of NDSS'15, 2015.

[8]. N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in Proc. Of ICML'16, vol. 48, 2016, pp. 201–210.

[9]. C. Dwork, "Differential privacy," in Proc. of ICALP'06. Springer, 2006, pp. 1–12.

[10]. K. Chaudhuri, A. D. Sarwate, and K. Sinha, "A near-optimal algorithm for differentially-private principal components." Journal of

Machine Learning Research, vol. 14, no. 1, pp. 2905–2943, 2013.

[11]. J. Zhang, Z. Zhang, X. Xiao, Y. Yang, and M. Winslett, "Functional mechanism: regression analysis under differential privacy," Proc. of VLDB'12, vol. 5, no. 11, pp. 1364–1375, 2012.

[12]. R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in Proc. of CCS'15. ACM, 2015, pp. 1310–1321.

[13]. M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in Proc. of CCS'16. ACM, 2016, pp. 308–318.

[14]. Y. Elmehdwi, B. K. Samanthula, and W. Jiang, "Secure k-nearest neighbor query over encrypted data in outsourced environments," in Proc. of ICDE'14. IEEE, 2014, pp. 664–675.

[15]. O. Goldreich, Foundations of cryptography: volume 2, basic applications. Cambridge university press, 2009.