TRANSACTIONS ON DATA PRIVACY 5 (2012) 297 — 331

Efficient Privacy Preserving Protocols for Similarity Join

Bilal Hawashin*, Farshad Fotouhi*,
Traian Marius Truta, and William Grosky*****
* Dept. of Computer Science, Wayne State University, Detroit, MI 48202
E-mail: {hawashin, fotouhi} @ wayne.edu
** Dept. of Computer Science, Northern Kentucky University,
Highland Heights, KY 41099, USA
E-mail: trutat1@nku.edu
*** Dept. of Computer and Information Science, University of Michigan - Dearborn
Dearborn, MI 48128, USA.
E-mail: wgrosky@umich.edu

Abstract. During the similarity join process, one or more sources may not allow sharing its data with other sources. In this case, a privacy preserving similarity join is required. We showed in our previous work [4] that using long attributes, such as paper abstracts, movie summaries, product descriptions, and user feedbacks, could improve the similarity join accuracy using supervised learning. However, the existing secure protocols for similarity join methods can not be used to join sources using these long attributes. Moreover, the majority of the existing privacy-preserving protocols do not consider the semantic similarities during the similarity join process. In this paper, we introduce a secure efficient protocol to semantically join sources when the join attributes are long attributes. We provide two secure protocols for both scenarios when a training set exists and when there is no available training set. Furthermore, we introduced the *multi-label supervised secure protocol* and *the expandable supervised secure protocol*. Results show that our protocols can efficiently join sources using the long attributes by considering the semantic relationships among the long string values. Therefore, it improves the overall secure similarity join performance.

1 Introduction

Similarity join consists of grouping pairs of records whose similarity is greater than a threshold, *T*. Privacy preserving protocols for similarity join are used to protect the data of two sources from being totally disclosed during the similarity join process. In the simplest case, when the join operation is done on two sources, A and B, source A is not

supposed to know the content of all the records in source B. Instead, source A can know the content of the records that will be joined with its own records. The records in source B that will not be joined with source A will be hidden from it, and vice versa. Even though similarity join have a wide range of applications in various fields, only a few researchers have concentrated on performing similarity join under privacy constraints.

Examples of such works includes [5], which introduced a protocol to perform similarity join using phonetic encodings, [3], which proposed a privacy preserving record matching protocol on both data and schema levels, [7], which concentrated on the e-health applications and its intrinsic private nature, and [17], which used a Term Frequency – Inverse Document Frequency (TF.IDF) based comparison function and a secure blocking schema. Other methods concentrated on using encryption to preserve privacy such as [1][2].

To our knowledge, the existing privacy preserving protocols for similarity join, such as [5][1][2][17], concentrated only on the syntax representation of the values, and did not consider the semantic relationships among them. Besides, in our previous work [4], we showed that the similarity join performance would be improved significantly when using long attributes as join attributes, instead of short attributes. However, the existing secure protocols are used mainly with short attributes. The term *short attribute* refers to any attribute of *short string* data type, whereas the term *short string* refers to the data type representing any string value of limited length, such as person name, paper title, and address. On the other hand, the term *long attribute* refers to any attribute of *long string* data type, whereas the term *long string* refers to the data type representing any string value with unlimited length, such as paper abstract, movie summary, and user comment.

In addition to the desired effect of improving the similarity join performance by using long attributes, long string attributes contain much more information than short string attributes, and most databases include such attributes. Furthermore, using long attributes can provide some level of privacy. The information to be shared can be written or merged as long strings without the need to include the secure data. In the following, we provide two applications illustrating the motivation to use long attributes in secure similarity join methods.

Example 1.1: Secure Semantic Similarity Join Protocol Using Long Attributes for Medical Data Integration

It is well known that medical records require a high level of privacy. All the reports, samples, diagnostics, and other medical related issues that belong to individuals in some medical center are protected from being disclosed. However, this could limit the usefulness of these records by adding constraints on sharing such information with

other medical centers. Furthermore, various medical centers could use different notations to refer to the same entity. In such case, using privacy preserving protocol for semantic similarity join is required. The privacy preserving part of the protocol is used to control the level of sharing and protect the secure data from being disclosed, and the similarity join part is used to resolve the different notations that refer to the same entity. Besides, using the long attributes and controlling its content by including only the information to be shared such as patient symptoms and diagnostics, and hiding the highly private data such as patient name and address, can add a level of privacy. Besides, sharing long attributes can increase the similarity join accuracy because such long fields provide more information than short ones.

Example 1.2: Secure Semantic Similarity Join Protocol Using Long Attributes for Car Companies
Recently, serious problems have been detected in some car designs that required call backs to such affected cars. Sharing a basic level of experience among car companies could be vital to avoid such scenarios in the future. Again, the various car companies use various jargons to refer to the same item. Moreover, the sharing process would be limited to the common interests only, and secure data in some company is not supposed to be disclosed to other companies. Therefore, a privacy preserving semantic similarity join protocol is needed. Commonly, the shared information, such as problem description and solution, could be written as long string values, without the need to write the secure information.

From the previous examples, it is worthwhile to study the use of long attributes to improve the similarity join performance and to consider the semantic relationships during the secure similarity join process. In our work, we consider both cases when a training set of similar records exists from various sources and when there is no available one.

The contributions of this work are as follows:

- We propose efficient secure protocols to perform similarity join when the join attribute is a long attribute, which can improve the secure similarity join accuracy.
- We find an existing method that can be used efficiently for joining values of long attributes under privacy constraints when no training set is available.
- We consider the semantic similarities among the string values during the secure similarity join process.
- Our protocols can assist the existing protocols, which are used mainly with short attributes, to improve the overall secure similarity join performance.

- We consider both cases when a training set exists and when there is no training set available, and solving both cases using our supervised and unsupervised secure similarity join protocols respectively.
- We consider both cases when each record refers to one or multiple entities at the same time.
- We propose a model for expandable supervised secure protocols.

The rest of this paper is organized as follows. Section 2 describes the candidate semantic methods to be compared with respect to joining long string values using similarity thresholds, when no training set is available. Section 3 describes our secure unsupervised protocol for semantic similarity join using long attributes and similarity thresholds. Section 4 compares the candidate semantic methods of section 2 and studies the performance of our secure unsupervised protocol when using the best performing methods from the previous comparison. Section 5 describes our secure supervised semantic similarity join protocol, when a training set exists with a single label per record. Section 6 discusses the multi-label supervised secure protocol, when a training set exists with multiple labels per record. Section 7 represents a model for the expandable supervised secure protocol. Section 8 is the conclusion

For our secure unsupervised protocol, when no training set is available, and in order to find the best semantic join method for long attributes, we compare diffusion maps [9], latent semantic indexing [10], and locality preserving projection [12]. These methods have strong theoretical foundations and have proved their superiority in many applications. Therefore, we compare their performance as candidate semantic similarity join methods for joining long attributes using similarity thresholds. Regarding the existing protocols, as stated earlier, they ignored the semantic relationships among the values by using merely phonetic encoding [5], encryption [1][2], or character-based distance methods [3]. Rationally, this limits the accuracy and causes a long running time when applied to long string values. Therefore, they are excluded from this comparison as they are not candidate semantic join methods for long atrributes. In order to evaluate the performance of our candidate semantic join methods for long attributes, we use two datasets, ***Amazon Products dataset*** [14] and ***IMDB Movies dataset*** [13]. After finding the best join method, we use various similarity threshold values to define the matched records and evaluate the unsupervised protocol.

For our secure supervised protocol, when a training set is available, and in order to find the best semantic join method for long attributes, which will be used later as part of our supervised protocol, we use the results of our work in [4], with some modifications to provide more privacy. In order to evaluate the performance of our supervised protocol, we use ***Pubmed dataset*** [23]. We use various classifiers from WEKA 3.6.2 to define the matched records and evaluate the supervised protocol after training them

using a small training set. Moreover, we use both single label and multi label classification in our supervised protocol.

It should be noted that in all our experiments, dataset preprocessing steps are conducted, which include removing stopwords, converting letters to small-letter case, and using full-word terms. Long string values are represented as vectors that are obtained upon applying the semantic method on the TF.IDF weighting vectors. More detailed explanation is given in Protocol 1 and Protocol 2.

2 Semantic Methods For Joining Long String Values Using Similarity Thresholds

In this section and the following two sections, we propose an efficient secure unsupervised similarity join protocol, when no training set exists. Such protocol is important when it is difficult or expensive to have a training set of similar records from various sources.

In order to capture the semantic relationships among the long string values, dimensionality reduction methods are used. Every dimensionality reduction method takes as an input the term by long string value matrix, where every row represents a term, and every column represents a long string value. The output of the dimensionality reduction method is reduced dimension by long string value matrix, where every row represents a reduced dimension, and every column represents a long string value. This step is done for all the records from all sources. Next, any two records from different sources are joined if the cosine similarity of their reduced representations is greater than a predefined threshold T. We compare several dimensionality reduction methods for joining long string attributes, and we will adopt the best method later in our unsupervised protocol. Hereafter, we describe the candidate dimensionality reduction methods.

2.1 Diffusion Maps

Diffusion maps are a dimensionality reduction method proposed by Lafon [9]. Initially, a weighted graph is constructed whose nodes are labeled with long string values and whose edge labels correspond to the similarity between the corresponding node values. A similarity function called the kernel function, W, is used for this purpose. The first-order neighborhood structure of the graph is constructed using a Markov matrix P. In order to find similarities among non-adjacent nodes, the forward running in time of a random walk is used. A Markov chain is computed for this purpose by raising the Markov matrix P to various integer powers. For instance, according to P_t, the t^{th} power of P, the similarity between two long string values x and y represents the probability of a

random walk from x to y in t time steps. Finally, the *Singular Value Decomposition* (SVD) dimensionality reduction function is used to find the eigenvectors and the corresponding eigenvalues of $P_{t,t\geq 1}$. The approximate pairwise long string value similarities are computed using the significant eigenvectors only. The similarity between any two long string values using such a method is called *diffusion maps similarity*. The mathematical details of diffusion maps are given below.

Consider a dataset V of N long string values, represented as vectors. Let x, y be any two vectors in V, $1 \leq x, y \leq N$. A weighted matrix $W_\sigma(x,y)$ can be constructed as

$$W_\sigma(x,y) = \exp(-\frac{D\cos(x,y)}{\sigma}), \quad (1)$$

where σ specifies the size of the neighborhoods that defines the local data geometry. As suggested in [11],

$$Dcos(x,y) = 1 - \frac{x.y}{\| x \| . \| y \|} . \quad (2)$$

We can create a new kernel as follows:

$$W_\sigma^\alpha(x,y) = \frac{W_\sigma(x,y)}{q_\sigma^\alpha(x) q_\sigma^\alpha(y)}, \quad (3)$$

where α deals with the influence of the density in the infinitesimal transitions of the diffusion, and

$$q_\sigma(x) = \sum_{y \in V} W_\sigma(x,y) . \quad (4)$$

Suppose

$$d_\sigma(x) = \sum_{y \in V} W_\sigma^\alpha(x,y) , \quad (5)$$

we can normalize the previous kernel to get an anisotropic transition kernel $p(x,y)$, as follows:

$$p_\sigma(x,y) = \frac{W_\sigma^\alpha(x,y)}{d_\sigma(x)}. \quad (6)$$

$p_\sigma(x,y)$ can be considered a transitional kernel of a Markov chain on V. The diffusion distance D_t between x and y at time t of the random walk is

$$D_t^2(x,y) = \sum_{z \in V} \frac{(p_t(x,z) - p_t(y,z))^2}{\phi_0(z)}, \quad (7)$$

where ϕ_0 is the stationary distribution of the Markov chain.

After using the SVD operation, the Markov chain eigenvalues and eigenvectors can be obtained. Therefore, the diffusion distance D_t can be written as:

$$D_t^2(x,y) = \sum_{j \geq 1}^{2t} \lambda_j (\varphi_j(x) - \varphi_j(y))^2. \quad (8)$$

We can reduce the number of dimensions by finding the summation up to a specific number of dimensions z. Thus, the mapping would be:

$$\omega : x \to (\lambda_1 \varphi_1(x), \lambda_2 \varphi_2(x), ..., \lambda_z \varphi_z(x)). \quad (9)$$

We set the values of σ and α to 10 and 1 respectively for our experiments, as used in [4].

2.2 Latent Semantic Indexing (LSI)

LSI [10] uses the Singular Value Decomposition operation to decompose the term by long string value matrix M, which contains terms (words) as rows and long string values as columns, into three matrices: T, a term by dimension matrix, S, a singular value matrix, and D, a long string value by dimension matrix. The original matrix can be obtained through matrix multiplication of TSD^T. In order to reduce the dimensionality, the three matrices are truncated to z user selected reduced dimensions. Dimensionality reduction reduces noise and reveals the latent semantics in the dataset. When the components are truncated to z dimensions, a reduced representation matrix, M_z is formed as

$$M_z = T_z S_z D_z^T. \quad (10)$$

Refer to [10] for a detailed explanation of this method.

2.3 Locality Preserving Projection

Locality preserving projection [12] is described briefly as follows. We are given a set of long string values represented in an m-dimensional vector space $x_1, x_2, x_3, \ldots, x_n$ in R_m, where m represents the dimensions. This method finds a transformation matrix A that maps these long values into $y_1, y_2, y_3, \ldots, y_n$ in a new reduced space R_z, $z<m$, such that $y_i = A^T x_i$. This method is particularly applicable when $x_1, x_2, x_3, \ldots, x_n \in O$, where O is a nonlinear manifold embedded in R_m. Refer to [12] for a detailed explanation of this method.

As stated at the beginning of this section, after applying every dimensionality reduction method, two long string values are joined if the cosine similarity of their reduced representations is greater than a threshold T. In section 4, we compare the previous methods as candidate semantic methods for joining long string attributes.

3 Secure Unsupervised Protocol For Semantic Similarity Join Using Long Attributes

In this section, our proposed secure unsupervised protocol is described. As stated before, this protocol is efficient in performing the similarity joining operation between two tables using their long string attributes when no training set is available. Besides, it considers the semantic relationships among the long string values. Up to our knowledge, no protocols have been proposed to be used with such long attributes, and as shown in [4], using such attributes provides a better semantic join accuracy than using short attributes.

In the protocol, we have two sources A and B, each of which has a relation, R_a and R_b respectively. First, the two sources share the similarity threshold value T that will be used later to decide similar pairs. Next, each source generates the term by long string value frequency matrix from its long attribute, such that each row represents a term (word), each column represents a long string value, and each cell value of row i and column j represents the frequency of the term i in the long string j. The result is M_a and M_b for A and B respectively. For example, if A contains 1000 paper abstract values in its Paper Abstract attribute, each row in M_a represents a term, each column represents an abstract, and each cell, which represents an intersection between a row and a column, represents the frequency of the corresponding term in the corresponding abstract. Later, the TF.IDF weighting is applied to both matrices. TF.IDF weighting is commonly used in information retrieval. TF.IDF weighting of a term w in a long string value x is given as follows:

$$\text{TF.IDF}(w,x)=\log(tf_{w,x}+1).\log(idf_w), \tag{11}$$

where $tf_{w,x}$ is the frequency of the term w in the long string value x, and idf_w is $\frac{N}{n_w}$, where N is the number of long string values in the relation, and n_w is the number of long string values in the relation that contains the term w.

Upon applying TF.IDF, both $WeightedM_a$ and $WeightedM_b$ are generated. Every row in this matrix represents a term, every column represents a long string value, and every entry represents the weight of the term in that long string value.

In the next step, both sources share the MeanTF.IDF threshold value [16] to be used and apply the MeanTF.IDF unsupervised feature selection method to both $WeightedM_a$ and $WeightedM_b$. This method assigns a numerical value for each term in both $WeightedM_a$ and $WeightedM_b$. The value of a term w is calculated as follows:

$$\text{Val}(w) = \frac{\sum_{x=1}^{N} TF.IDF(w,x)}{N}, \tag{12}$$

where TF.IDF(w,x) is the TF.IDF weight of the term w in the long string value x, and N represents the number of long string values in the relation. The value of each term represents its importance. The terms with the highest values are the most important terms. It should be noted that terms and features are used alternatively in this context and have the same meaning.

The features from A with the highest values are concatenated with randomly generated features by A and are sent to a third source, C. B does the same. C does not provide information, but it serves as an intermediate part to process the data sent from both A and B. C finds the intersection between the two feature sets and returns those shared features, SF, that exist in both sources. Both sources remove their randomly generated features from SF and generate new matrices, SF_a and SF_b, where each row represents an important term from SF, each column represents a long string value, and each entry represents the TF.IDF weighting. Later, every source adds random records to its corresponding matrix to hide its original data. It should be noted that in this step, every record, including the randomly generated ones, is assigned a random index number. The generated matrices, *Rand_Weighted_a* and *Rand_Weighted_b* are sorted according to their index number to guarantee that the randomly generated records are randomly distributed in both matrices. Next, both matrices are sent to C. C performs the semantic operation on both matrices to produce *Red_Rand_Weighted_a* and *Red_Rand_Weighted_b*.These matrices have the concept terms as rows and the long string values as columns. In section four of the paper, we will compare different candidate semantic methods when various thresholds are used, and the best method will be used

there. Also, we will study the effect of adding random records on the semantic operation performance in that section. The protocol continues by finding the cosine similarities for all the pairs (x,y), $x \in$ *Red_Rand_Weighted_a* and $y \in$ *Red_Rand_Weighted_b*, and if the cosine similarity is greater than a threshold T, the pair of the two vectors is considered a match and inserted into *Matched*. *Matched* is returned to both A and B to delete the pairs that include randomly generated records. Finally, both sources can share their Matched list after deleting the random records.

In this previous protocol and similar ones, two types of errors exist. Including non-similar pairs in the *Matched* set (false positives), and excluding similar pairs from the *Matched* set (false negatives). Every domain is different in handling such errors. If solving the false negatives is more important than false positives, the similarity threshold T in line 1 of protocol 1 could be decreased to include in the *Matched* set all the actual similar pairs. This would decrease the number of missed pairs to the minimum. However, this would introduce more incorrect similar pairs in the *Matched* set. This could be eliminated manually or left unchanged. If solving the false positives is more important, the similarity threshold T could be increased accordingly. Again, this would increase the number of false negatives. A compromised value for T would provide a sufficient solution for both types of errors.

In this protocol we assume that the similarity join process is done using one join attribute only. When the join process is done using multiple join attributes simultaneously, the protocol is used for every long string attribute alone, and the intersection among the Matched sets, which are found in line 15 of Protocol 1, could be used to get the final set of Matched records.

One issue with the protocol is having a randomly generated feature in the returned *SF*. This could occur when the two sources generate randomly the same feature, or when one source generates a random feature that matches an important feature in the other source. In order to calculate the probability of such scenarios, we assume that the randomly generated strings have length up to k characters. For a specific length s, the number of generated strings is 26^s for English alphabet. Therefore, the probability of generating a string that matches with an existing feature is

$$P = \frac{1}{\sum_{s=1}^{k} 26^s}, \tag{13}$$

and the probability of generating the same random feature by both sources is P^2.

For example, if we generate random strings of lengths up to 5 characters, the probability of the first scenario will be around 10^{-19}, and the probability of the second one is 10^{-38}, which are very unlikely. Furthermore, these scenarios will not affect the

running of the protocol, but will make SF_a and SF_b different in the number of rows (features). However, adding a few features to one matrix will not affect significantly the results because we use only the main eigenvectors and eigenvalues in the semantic methods.

4 Experiments

In order to evaluate the previous semantic methods in joining long string values when no training set is available, two datasets were used, Amazon Products Dataset and the IMDB Movies Dataset. Table 1 below describes the use of these datasets.

The following is a brief description of each dataset.

Amazon Products

We collected 700 records from the Amazon website via http://amazon.com. In this work, we are interested in the product descriptions, which provide detailed information about the products. The product descriptions were divided into categories, such as computers, perfumes, cars, and so on. All product descriptions that belong to the same category are considered similar. The total number of categories in the collected dataset is 13 categories. The categories of the collected descriptions were of various complexities, and the number of records in all categories is approximately equal.

Internet Movies Database (IMDB)

We collected 1000 records from the IMDB Movies database, which is available online via http://imdb.com. Typically, every movie has multiple summaries, written by various users. All summaries that belong to the same movie are considered of the same category. The total number of categories in the collected dataset is 10 categories. As in the previous dataset, the categories of the collected movies were of various complexities, and the number of records in each category is approximately equal.

Protocol 1: Secure Unsupervised Protocol for Semantic Similarity Join using Long Attributes
Input: Two sources ***A*** and ***B***, each has a long attribute ***X***.
Output: Set of matched records sent to both ***A*** and ***B***.
Protocol:

(1) Both ***A*** and ***B*** share the similarity threshold ***T*** to decide matched pairs.

(2) ***A*** and ***B*** generate their term by long string value matrices M_a and M_b from $R_a.X$ and $R_b.X$.

(3) TF.IDF weighting is calculated from M_a and M_b to generate $WeightedM_a$ and $WeightedM_b$.

(4) Both ***A*** and ***B*** share the MeanTF.IDF threshold value to perform MeanTF.IDF unsupervised feature selection.

(5) Both ***A*** and ***B*** return their selected features along with some randomly generated features to a third source ***C***.

(6) ***C*** finds the shared features in both sources, ***SF***, and returns them to both ***A*** and ***B***.

(7) ***A*** and ***B*** generate reduced weighted matrices SF_a and SF_b from $WeightedM_a$ and $WeightedM_b$ using ***SF*** after removing the randomly generated features.

(8) ***A*** generates random records, each of which has SF entries and add them randomly to SF_a. ***B*** does similarily.

(9) Every origional and random record in both SF_a and SF_b is assigned a random index number, and both sources keep track of the index numbers that belong to the randomly generated records.

(10) Both SF_a and SF_b are sorted according to the index number to generate ***Rand_Weighted_a*** and ***Rand_Weighted_b***, which are sent later to ***C***.

(11) ***C*** performs the semantic operation to generate ***Red_Rand_Weighted_a*** and ***Red_Rand_Weighted_b***.

(12) ***C*** finds the pairwise cosine similarities among the generated two matrices.

(13) If the cosine similarity for a pair is greater than the predefined threshold ***T***, this pair is inserted into ***Matched***.

(14) ***C*** returns ***Matched*** to both ***A*** and ***B***.

(15) Both ***A*** and ***B*** delete from ***Matched*** the randomly generated records.

(16) ***A*** and ***B*** share their ***Matched*** lists.

It should be noted that dividing each dataset into parts to represent the data from different sources would not make any difference from using the whole dataset as one part, and this is due to the diffusion maps kernel, which requires the pairwise distance matrix for the records from all sources. This step would be the same if the whole dataset is used. Later, the data is divided by the intermediate source C as stated in line 11 of Protocol 1.

For our experiments, we used an Intel® Xeon® server with 3.16GHz CPU and 2GB RAM, with Microsoft Windows Server 2003 Operating System. Also, we used Microsoft Visual Studio 6.0 to read the datasets, and Matlab 2008a for the implementations of the candidate semantic methods. For diffusion maps, we used the Lafon implementation [9]. Regarding LSI, we used the Matlab svds( ) operation, and for locality preserving projection, we used the implementation provided in [15].

Table 1. Datasets Description

Dataset	Used Number of Records	Number of Categories
Amazon Products	700	13
IMDB (Movies)	1000	10

In order to evaluate the performance, we used F_1 measurement, *preprocessing time*, *operation time*, and *matching time*. They are defined as follows:

- F_1 rating is the harmonic mean of the method *recall* and the *precision*. It is given as

$$F_1=\frac{2*R*P}{R+P}, \tag{14}$$

where R represents the recall, which is the ratio of the relevant data among the retrieved data, and P represents the precision, which is the ratio of the accurate data among the retrieved data. Their equations are given as follow:

$$R = \frac{TP}{TP+FP}, \text{ if } TP+FN>0,$$
otherwise undefined. (15)

$$P = \frac{TP}{TP+FN}, \text{ if } TP+FN>0,$$
otherwise undefined. (16)

In order to find these measurements, a two-by-two contingency table is used for each category. Table 2 below represents this contingency table.

- Preprocessing time is the time needed to read the dataset and generate matrices that are used later as an input to the semantic operation.
- Operation time is the time needed to apply the semantic method.
- Matching time is the time required by the third source, *C*, to find the cosine similarity among the records provided by both *A* and *B* in the reduced space and to compare the similarities with the predefined similarity threshold.

Table 2. The Contingency Table to Describe the Components of the Recall and Precision

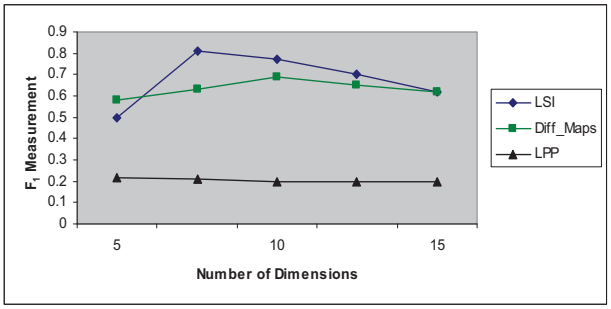
Actual Class	**Predicted Class**		
		Class = Yes	*Class = No*
	Class = Yes	TP	FN
	Class = No	FP	TN

Before starting our experiments, we used the TF.IDF with Cosine Similarity as a baseline semantic method for joining long string attributes. A term by long string value matrix was used, where every row represents a term, every column represents a long string value, and every cell of row *i* and column *j* represents the TF.IDF weighting of the term *i* in the long string value *j*. As a baseline method, neither feature selection methods nor dimensionality reduction methods described later were used here. In order to semantically join two long string values according to this method, the cosine similarity was used with a predefined threshold. The used threshold values varied between 0.3 and 0.9. The experimental results showed a very poor performance on both IMDB and Amazon Products datasets. The maximum F_1 measurement was 0.05 and 0.04 respectively. The recall values were very high, and the precision values were very low. We conducted other set of experiments to study the effect of using feature selection methods alone without any dimensionality reduction method. We used MeanTF.IDF feature selection method and selected the most important 2% terms. The maximum F_1 measurement was 0.63 and 0.33 in IMDB and Amazon Products datasets respectively. We believe that this performance could be enhanced more by using dimensionality reduction methods because of their ability to extract the semantic relationships among the long string values. Besides, reducing the dimensions would decrease the algorithm running time. As a preliminary work, we used Diffusion Maps dimensionality reduction method after using the MeanTF.IDF. The obtained F_1 measurement increased to 0.71 and 0.52 in IMDB and Amazon Products datasets respectively. This introduces our work.

Our work in this section is divided into two phases. In Phase 1, we want to find the best semantic candidate method to join long string values when similarity thresholds are used. Later, for Phase 2, we will use as part of our protocol the best semantic method

from Phase 1. For Phase 1, we compared diffusion maps, latent semantic indexing, and locality preserving projection. As every method is a dimensionality reduction method, we used the optimal number of dimensions for each method that maximizes the F_1 measurement. Figure 1 shows an example of selecting the optimal number of dimensions experimentally on IMDB dataset. In this Figure, we found the F_1 measurement for various numbers of dimensions ranging from 5 to 25. We used a fixed similarity threshold value in this case, which was 0.7. Obviously, the maximum F_1 measurement was obtained using ten dimensions. The optimal number of dimensions for the remaining methods was calculated similarly. The best number of dimensions for LSI was eight, while it was five for LPP. It should be noted that the given optimal numbers of dimensions are data-dependent. Using different dataset or even different data from the same dataset could lead to different optimal numbers. Figure 2 depicts the comparison of the three methods using various similarity thresholds on IMDB dataset. According to the Figure, both LSI and Diffusion Maps worked efficiently, with advantage given to LSI. The maximum F_1 measurement for LSI was 0.81, with threshold 0.7, while the maximum F_1 measurement for Diffusion Maps was 0.71, with threshold 0.5. Locality Preserving Projection showed the worst performance.

Figure 1. Finding best number of dimensions for diffusion maps, LSI, and LPP experimentally. IMDB dataset was used. The best numbers of dimensions were ten, eight, and five dimensions respectively, which resulted in the highest F_1 Measurements.



For Amazon Products dataset, Figure 3 displays the results. Clearly, diffusion maps and LSI outperformed LPP. The performance of LSI dropped significantly in this dataset in comparison with diffusion maps. We concluded from Phase 1 that both diffusion maps and LSI showed efficient performance in joining long string values with advantage given to diffusion maps due to its stable performance.

Figure 2. Comparing LSI, diffusion maps, and locality preserving projection to find the best semantic method for long attributes. IMDB dataset was used. Both LSI and diffusion maps showed the best performance.

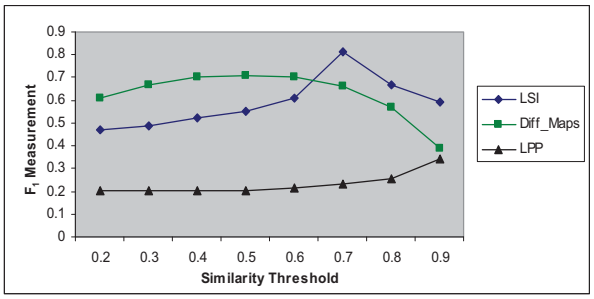
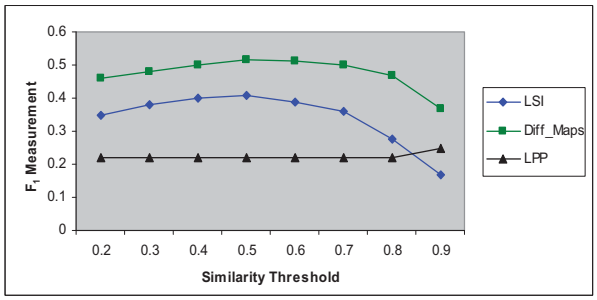


Figure 3. Comparing LSI, diffusion maps, and locality preserving projection to find the best semantic method for long attributes. Amazon Products dataset was used. Diffusion maps showed the best performance.



In Phase 2 of the work in this section, we used diffusion maps and LSI, as they showed the best performance in Phase 1. We used them separately with our protocol and studied the protocol performance. We used both IMDB and Amazon datasets in this phase. The evaluation measurements used here are preprocessing time, operation time, and matching time. It should be noted that the F_1 measurement for both methods was studied in Phase 1, where both methods showed efficient performance with advantage given to diffusion maps.

Regarding the preprocessing time, it took 12 seconds to read 1000 records from IMDB, while it took one second to find TF.IDF weighting, and 0.5 second to apply MeanTF.IDF. Time to find shared features by A and B was negligible (approximately zero). For Amazon Products dataset, similar trends were found.

For operation time, Figure 4 represents the results for LSI and diffusion maps with various dimensions in both IMDB and Amazon Products datasets. Obviously, the time needed to perform LSI is less than that in Diffusion Maps. The difference increases with

the increase in the number of dimensions. For Amazon Products dataset, similar trends were found.

It is worthwhile to mention that this operation is done once only, and therefore, does not highly affect the protocol performance. Also, it is not necessary to have a large number of dimensions for diffusion maps to get the optimal performance. The optimal number of dimensions for diffusion maps in the IMDB dataset was ten, while it was five for Amazon Products dataset.

Regarding the matching time, and due to the small number of dimensions used to represent each long string value, this time was negligible, even with the Cartesian product comparison of 5000 records. For the Amazon Products dataset, similar trends were found.

Moreover, we studied the effect of adding random records, as stated in steps 8-10 in the protocol, on the performance of the semantic operation, which is done in step 11. We added various portions of random records that are dataset size dependent, and we found their effect on both the F_1 measurement and the number of suggested matches. Regarding F_1 measurement, the performance increased slightly when a small number of random records were added, then it started to decrease. This is due to the mechanism of the semantic operation itself. In diffusion maps, the important eigenvalues and eigenvectors are extracted from the dataset. The more random records are inserted, the more their effect on the real eigenvectors and eigenvalues. At some point, the protocol will extract eigenvector(s) and eigenvalue(s) that represent the random records, which will decrease the accuracy significantly. Figure 5 depicts this step. What was not expected is the slight enhancement in the accuracy upon adding a small number of random records. Theoretically, adding random records will increase the number of features in the kernel matrix, which can make the categories more separable and increase the classification accuracy providing that the eigenvectors are not affected by the added noise. Regarding the number of suggested matches, trivially, increasing the number of records by adding random records will increase the number of candidate pairs, which in turn will increase the suggested matches. Adding random records will increase the number of candidate pairs to be compared, which will increase the number of suggested matches. Adding more random records will consume more time and place more overhead. Figure 6 illustrates this step. Overall, we conclude that adding random records which compose 10% of the whole data size will hide the real data, without much effect on both the semantic operation accuracy and running overhead.

Figure 4. Operation Time for Diffusion Maps and LSI with various number of dimensions. IMDB and Amazon Products datasets were used. Operation time for LSI was less than that in diffusion maps.

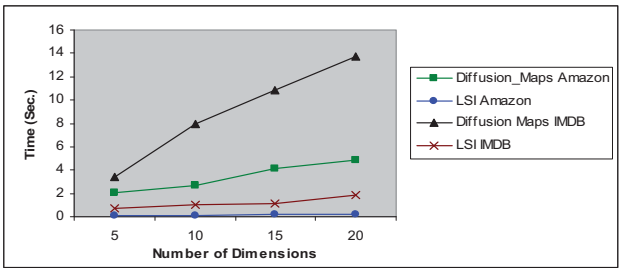


Figure 5. The effect of adding random records on the F_1 measurement upon using diffusion maps. F_1 measurement decreased rapily when the inserted random records size exceedes 10% of the dataset size.

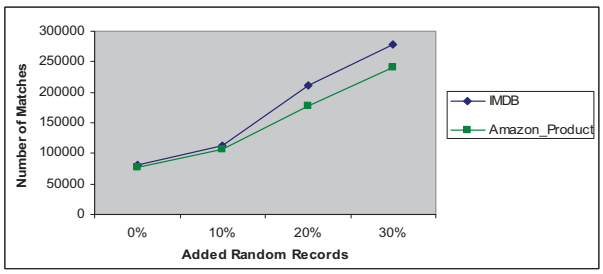
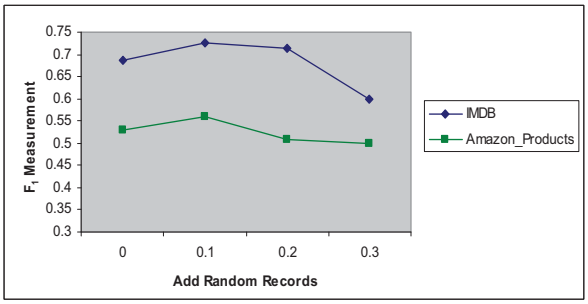


Figure 6. The effect of adding random records on the number of suggested matches upon using diffusion maps. Adding more reocrds introduced more overhead by increasing the number of suggested matched records.



5 Secure Semantic Similarity Protocol for Long Attributes using Supervised Learning

In the previous sections, we proposed a secure semantic similarity join protocol for long string attributes using similarity thresholds. While this protocol is simple, always applicable, and efficient, the performance could be enhanced even further when training set is used. Using such a training set could improve the similarity join performance because it reflects the user preferences and would train the classifier to label the testing records accordingly. It could be difficult to have a large training set; however, a small representative training set could be created without much human effort. This would not be obtained in the case of unsupervised learning. Furthermore, using distance methods are absolute and domain independent; however, in real life applications, classification labels are domain dependent, and what is similar in some domain could be considered not similar in another domain. For example, a laptop and a desktop computers could be considered similar in some domains; however, they could be considered non similar in other domains. While hierarchical clustering can help in this case, it uses absolute distance methods. The similarity join process is supposed to join records according to the user preference. In conclusion, it is worthwhile to use a small training set in the semantic similarity join process.

Although many previous works have studied the semantic similarity join process using supervised methods, when a training set exists [18][19][20], to our knowledge, no work has proposed a secure protocol for such process. As stated before, using such a protocol would improve the semantic similarity join performance and provide domain dependent solutions under security constraints. We illustrate one example of using such a secure supervised protocol as follows.

Example 5.1: The Use of Supervised Secure Similarity Join for Disease Detection
As more and more diseases are discovered, a level of experience sharing among medical centers is highly recommended to diagnose the existing diseases precisely and according to the recent discoveries. As stated before, the information to be shared could be represented as long attributes. In order to share experience among various centers, the already diagnosed diseases in every medical center could serve as a training set. For example, records of diseases symptoms and diagnosis could be represented in two long string attributes and serve as a training set, one record per disease. A unified labeling policy is required in this case to ensure that the label has the same meaning in different medical locations. In this example, the disease name will serve as the record label. Next, the secure semantic similarity join protocol for long attributes using supervised methods could be used to provide the information integration. Later, any new case in some

medical center could be diagnosed using the shared experience, and the best diagnosis will be provided.

In this section and the following two sections, we propose a secure supervised protocol for semantic similarity join using long attributes. In order to find the best semantic method to join long attributes using supervised learning, we introduce hereafter our previous work in improving similarity join performance using diffusion maps and long string attributes using supervised learning[4]. Later, we discuss our supervised similarity join secure protocol, which is a modification of our algorithm, to provide more privacy.

5.1 Improving the Similarity Join Performance Using Diffusion Maps and Long String Attributes Using Supervised Learning

In our previous work in [4], we proposed using long string attributes as join attributes to improve the semantic similarity join performance using supervised learning, when a training set exists. However, we did not consider the privacy issue. In that work, we compared Diffusion Maps [9], LSI [10], Eigenvectors [21], SoftTfIdf with Cosine Similarity [22], and TF.IDF with Cosine Similarity in order to find the best similarity method for long attributes. We used a Pubmed Dataset [23]. We used both the SVM classifier and Bagging learning methods. The results showed that Diffusion Maps was the best among compared for joining long string values using supervised learning. Therefore, we are going to adopt the use of Diffusion Maps as the similarity method for long attributes using supervised learning. In that Diffusion Maps based Semantic Similarity Join algorithm, the classifier training and testing sets represent each record in the diffusion maps reduced form. We used the equations in [4] to convert each testing record into this reduced form. For more information about our previous work, please refer to [4].

The following is a brief description of the Pubmed Dataset, which was used in our previous work, and will be used to evaluate the performance of our secure supervised similarity join protocol.

5.2 Pubmed Dataset

This dataset includes indexed bibliographic medical citations and abstracts. It is collected by the U.S. National Library of Medicine (NLM). It includes references from more than 4500 journals. The total number of categories is 23 classes proposed by [24]. Appendix A lists a description of the 23 classes. In our experiments, we used subsets of various numbers of records and numbers of categories. As a long string attribute, we used the paper abstract. PUBMED citations and abstracts could be accessed by

PUBMED via http://www.ncbi.nlm.nih.gov/entrez/query.fcgi or by the NLM Gateway via http://gateway.nlm.nih.gov/gw/Cmd.

As stated in section 4, dividing the dataset into parts to simulate the data of different sources would not make any difference from using the single undivided dataset, and this is because of the diffusion maps kernel, which requires grouping the records from all sources to find the pairwise distance among them. This would produce the same result as using a single undivided dataset. Besides, in the supervised learning context, there is no need to divide the data later because it will serve as a single training set for the testing records from all the sources.

5.3 Our Secure Semantic Similarity Join Protocol Using Long Attributes Using Supervised Learning

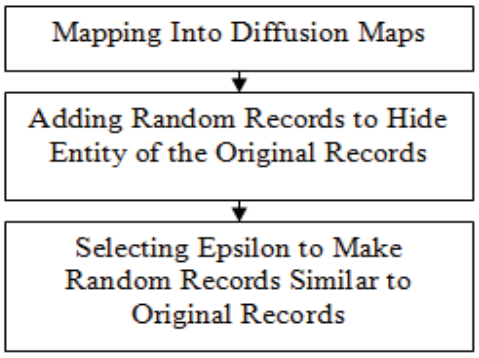
From our previous work in [4], and as stated in section 5.1, it is clear that Diffusion Maps is one of the best methods to be used with long attributes using supervised learning methods. However, we need to modify the method to provide more privacy. Initially, such a method provides a level of privacy by mapping the data into the diffusion maps space. In order to increase the privacy level, random records are to be added by every source before the sharing process in order to hide the original data. However, using pure random records could be inefficient as it is easy to detect. Our faked record should be as close to an original looking record as possible, in order to make it harder to be detected. Moreover, having pure random records in the training set and assigning them to random labels would affect the classifier learning model and decrease the classification F_1 measurement when a testing set is used. Therefore, the added records need to be carefully selected to provide a privacy level and to protect the classification accuracy from being decreased. Our selection method is described next.

In order to generate each random record in some source, the source needs to pick a record from its original records randomly and change each value randomly. Epsilon is used as an upper limit to the change in each value. The equation to generate a random record vector of n values from an existing record is the following.

$$\text{Random\_Vector}(i) = \text{Existing\_Vector}(i) +/- \text{Epsilon}, \quad (17)$$

where i=1:n, Epsilon is a user defined value representing the maximum value change, the sign is selected to be positive or negative randomly, and Existing_Vector is the randomly selected original record. It should be noted that a different existing original record is selected as a seed for each new Random_Vector.

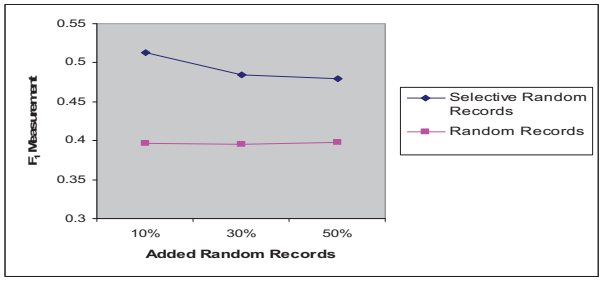
Figure 7 . The Privacy Layers of our Supervised Protocols.



In order to study the effect of adding selective random records instead of pure random records, we used both methods with various noise portions ranging from 10% to 50% added to the training set, and we used the SVM classifier later to classify a testing set using that training set. As displayed in Fig. 8, using selective random records preserved the classifier accuracy, in contrast with the pure random records, which decreased the classification accuracy significantly. Therefore, we adopted the use of selective random records in the following experiments.

Regarding the effect of adding selective random records instead of pure random records on privacy, and as explained before, using such selective random records would make it harder to be detected and distinguished from the original records, and this improves the level of privacy accordingly. Fig. 7 depicts the privacy layers of our secure supervised protocol. In the top layer, mapping into diffusion maps space provides the first level of privacy. Next, adding random records to the original records from each source would provide other level of privacy by hiding the entities of the original records. Finally, processing the random records to make them selective random records using the epsilon value provides the third level of privacy. It should be noted that our secure unsupervised protocol contains the top two layers only because it uses pure random records. Using selective random records in our secure unsupervised protocol would improve its privacy, and studying the effect of selective random records on that protocol is left to the future work.

Figure 8. Comparing Selective Random Records with Random Records. Clearly, using selective random records achieved more F_1 measurement.



The Secure Similarity Join Protocol for Long Attributes Using Supervised Learning is as follows.

In Protocol 2, which states our secure protocol for semantic similarity join for long attributes using supervised learning, we have two sources A and B, each of which has a relation, R_a and R_b respectively. First, each source generates the term by long string value matrix from its long attribute X, such that each row represents a term (word), each column represents a long string value, and each cell value, which is the intersection of row i and column j, represents the frequency of term i in the long string value j. The result is M_a and M_b for A and B respectively. For example, if A contains 1000 Disease Descriptions in its Disease Description attribute X, each row in M_a represents a term, each column represents a disease, and each cell value represents the frequency of the term in the disease description. Later, the TF.IDF weighting is applied to both matrices. TF.IDF weighting was already described in Equation 11.

Upon applying TF.IDF, both $WeightedM_a$ and $WeightedM_b$ are generated. As in protocol 1, every row in this matrix represents a term, every column represents a long string value, and every entry represents the weight of the term in that long string value.

In the next step, both sources share the χ^2 threshold value [25] to be used and apply the χ^2 supervised feature selection method to both $WeightedM_a$ and $WeightedM_b$, as used in our previous work [4]. This method assigns a numerical value for each term in both $WeightedM_a$ and $WeightedM_b$. The value of a term w is calculated as follows:

$$\text{Val(w)} = \sqrt{\frac{(n_{pt+} + n_{nt+} + n_{pt-} + n_{nt-})(n_{pt+}n_{nt-} - n_{pt-}n_{nt+})^2}{(n_{pt+} + n_{pt\_})(n_{nt+} + n_{nt-})(n_{pt+} + n_{nt+})(n_{pt-} + n_{nt\_})}}\ , \tag{18}$$

where $\mathbf{n_{pt+}}$ and $\mathbf{n_{nt+}}$ are the number of documents in the positive category and the negative category respectively in which term **w** appears at least once. The positive and negative categories are used to find the accuracy measurements per class when multiple classes are used such that the positive category indicates a class and the negative

category indicates the remaining classes. $\mathbf{n}_{pt-}$ and $\mathbf{n}_{nt-}$ are the number of documents in the positive category and the negative category respectively in which the term $\boldsymbol{w}$ doesn't occur. The value of each term represents its importance. The terms with the highest values are the most important terms.

The features from *A* with the highest values are concatenated with randomly generated features by *A* and are sent to a third source, *C*. *B* does the same. Later, *C* finds the intersection and returns those shared features, *SF*, that exist in both sources. Both sources remove their randomly generated features from *SF* and generate new matrices, SF_a and SF_b, where each row represents an important term from *SF*, each column represents a long string value, and each entry represents the TF.IDF weighting. Later, every source adds selective random records to its corresponding matrix to hide its original data. The records are generated using Equation 17 as described previously. Again, every record, including the randomly generated ones, is assigned a random index number. The generated matrices, *Rand_Weighted_a* and *Rand_Weighted_b* are sorted according to their index number to guarantee that the randomly generated records are randomly distributed in both matrices. Next, both matrices are sent to *C*. *C* performs the semantic method, which is the Diffusion Maps as suggested in [4] for joining long string values using supervised learning. Applying the semantic method on both A and B produces *Red_Rand_Weighted_a* and *Red_Rand_Weighted_b*. These matrices have the concept terms as rows and the long string values as columns. In our experiments, we used Diffusion Maps based semantic join as described in section 5.1. Later in this section, we will conduct more experiments to study the effect of adding selective random records and changing epsilon value on the semantic operation performance. The protocol continues by training a classifier using all the pairs (x,y), $x \in$ *Red_Rand_Weighted_a* and $y \in$ *Red_Rand_Weighted_b*. Again, one major difference between this protocol and Protocol 1 is that every long string value in the attribute *X* in Protocol 2 has a label that refers to its category, in contrast with Protocol 1 that manipulates unlabeled long string values. Upon training the classifier, *A* sends its testing records, along with some random records, to *C* for classification. C classifies the records and returns their predictions. *B* sends its testing records similarly. After excluding the random records, both *A* and *B* shares the labels of their original records, and the original records belonging to the intersected labels are shared between the two sources.

In this section, we used the Pubmed medical dataset to evaluate the protocol performance. Besides, we used an Intel® Xeon® server of 3.16GHz CPU and 2GB RAM, with Microsoft Windows Server 2003 Operating System. Also, we used Microsoft Visual Studio 6.0 to read the datasets, Matlab 2008a for the implementation of the Diffusion Maps, and Weka 3.6.2 for the SVM classifier to get the method's performance.

Protocol 2: Secure Supervised Protocol for Semantic Similarity Join using Long Attributes

Input: Two sources ***A*** and ***B***, each has a long attribute ***X***. A small training set ***TR*** that has labeled long string values.

Output: Set of matched records sent to both ***A*** and ***B***.

Protocol:

(1) ***A*** and ***B*** generate their term by long string value matrices M_a and M_b from TR_a and TR_b.

(2) TF.IDF weighting is calculated from M_a and M_b to generate $WeightedM_a$ and $WeightedM_b$.

(3) Both ***A*** and ***B*** share the χ^2 supervised feature selection threshold, and each source performs χ^2 on its own terms.

(4) Both ***A*** and ***B*** return their selected features along with some randomly generated features to a third source ***C***.

(5) ***C*** finds the shared features in both sources, ***SF***, and returns them to both ***A*** and ***B***.

(6) ***A*** and ***B*** generate reduced weighted matrices SF_a and SF_b from $WeightedM_a$ and $WeightedM_b$ using ***SF*** after removing the randomly generated features.

(7) ***A*** generates selective random records, each of which has **SF** entries using Equation 17 and adds them randomly to SF_a. ***B*** does similarily.

(8) Every original and random record in both SF_a and SF_b Is assigned a random index number, and both sources keep track of the index numbers that belong to the selective random generated records.

(9) Both SF_a and SF_b are sorted according to the index number to generate ***Rand_Weighted_a*** and ***Rand_Weighted_b***, which are sent later to ***C***.

(10) ***C*** performs the semantic operation to generate ***Red_Rand_Weighted_a*** and ***Red_Rand_Weighted_b***.

(11) ***C*** trains a classifier on the training set which is composed of ***Red_Rand_Weighted_a*** and ***Red_Rand_Weighted_b***

(12) Both ***A*** and ***B*** sends their ***X*** long values, after converting them to a suitable form as discussed in [26] , into ***C*** for classification. Random records are sent also to ***C*** to hide the original entities.

(13) ***C*** classifies the records of ***A*** and returns the labels back. ***C*** classifies the records of ***B*** similarily.

(14) ***A*** deletes the random records and extract the labels of the original records in ***X***. ***B*** does similarly.

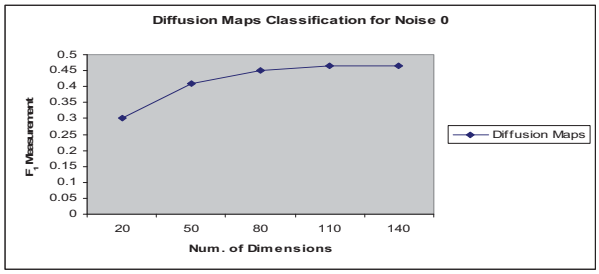
(15) ***A*** and ***B*** send their labels to ***C,*** and ***C*** returns the shared labels to both ***A*** and ***B***.

(16) ***A*** and ***B*** share the original records that belongs to the shared labels.

In order to evaluate the performance of Diffusion Maps as a semantic method, we used the same performance measurements used in [4], which are F_1 measurement, operation time, *classifier training time*, and *classifier testing time*. Both F_1 measurement and operation time were already defined in section 4. Classifier training time is the time needed by the classifier to create a model, and classifier testing time is the time needed by the classifier to label every testing record.

Initially, we labeled a subset of 816 records manually, and used them as a small labeled dataset, which includes 17 disease classes. Besides, every record was allowed to have single label only. In order to find the best diffusion maps reduced number of dimensions, we used various dimensions and we calculated the corresponding F_1 measurement. No noise was added in this phase, as this was done in the single source level, where no privacy was needed. We used Weka SVM Classifier and 10-Fold Cross Validation in order to get the F_1 measurement, which is described in Section 4. The optimal number of dimensions in our experiments was eighty, as the F_1 measurement tends to be stable after this value. Fig. 9 depicts the results.

Figure 9. Selecting the optimal number of diffusion maps reduced dimensions. SVM with 10-Fold cross validation was used on a subset of Pubmed containing 816 records. Eighty dimensions were selected as the performance becomes stable after that number.



Regarding the preprocessing time, it took three seconds to read the 816 training records from Pubmed, while it took negligible time to find TF.IDF weighting, and 204 second to apply χ^2. Time to find shared features by A and B was negligible (approximately zero).

For operation time, diffusion maps required three seconds, while for training time, it took 12 seconds to train the SVM classifier using 80 dimensions and 10 folds cross validation.

In the next step, we studied the effect of adding noise to the classification performance. We added various noise percentages from 10% to 30% to the training set, and we used various epsilon values from 1 to 100 to generate the random records. The SVM classifier was used to classify 4000 testing records using that modified training set. Figure 10 summarizes the findings.

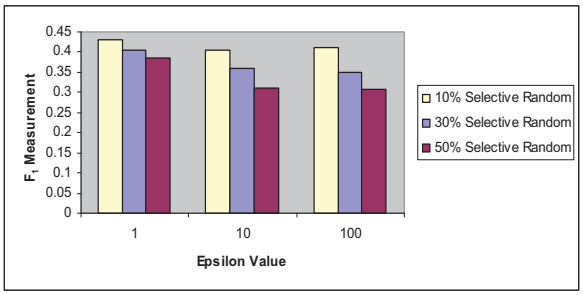
Obviously, increasing the noise percentage can provide more privacy but it would decrease the F_1 measurement. This is reasonable because it is affecting the SVM training model. Likewise, increasing the epsilon value (up to a maximum limit) would improve the privacy and decrease the F_1 measurement. However, a large increase in epsilon value would have negative effect on the privacy, because the records will be easy to detect and excluded as faked. Selecting the noise percentage and epsilon value is domain dependant and depends on the application privacy requirements versus the join accuracy.

Protocol3: Secure Protocol for Semantic Similarity Join using Long Attributes for Supervised Learning.
Input: A new test case arriving a source.
Output: Classifying this test case to the up to date knowledge and joining it according to semantic similarity.
Protocol:
(1) The training model is sent to both A and B.

(2) For every new test case arriving any source, the training model is used to classify it.

(3) This test case is joined to the shared records of its same label.

Figure 10. The effect of adding selective random records and changing Epsilon value on the F_1 measurment upon using diffusion maps. F_1 measurement decreased as the selective random records portion and epsilon value increase.

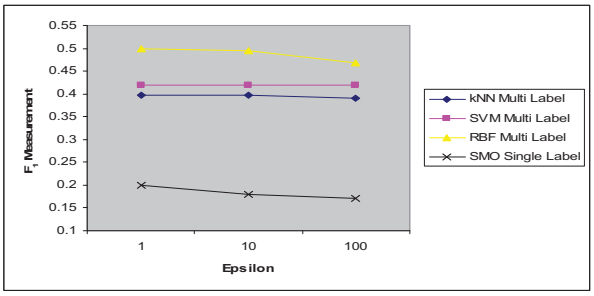


Besides, it is worthwhile mentioning that it took two seconds to read the 4000 testing records and negligible time to find the TF.IDF weighting. For the classifier testing time, it took two seconds to classify those testing records using the previous training set of 816 records.

6 Secure Semantic Similarity Join Protocol For Long Attributes Using Supervised Learning For Multi-Label Scenario

One limitation of the previous supervised solution is its constraint on the number of labels per record. So far, we forced every record in the training and testing sets to have one label. However, this is not always correct. In many real life applications, a record can belong to various entities and refer to multiple labels simultaneously. For example, a disease could be a virus disease (Label 1) and affect infants only (Label 2). Using multi-label classification would provide a model which is closer to real-life applications.

Figure 11. Comparing various multi-label classifiers with a single label classifier using various Epsilon values. Multi-label classification significantly outperformed single-label classification, and RBF Network classifier has the best F_1 measurement. 10% Noise was used.



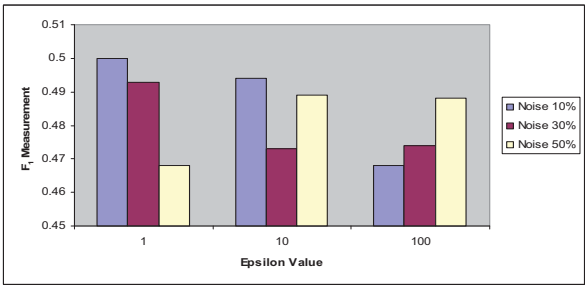
Again, to our knowledge, no work has been done to benefit from multi-label classification techniques in the secure supervised protocol for similarity join. Therefore, we studied the performance of various multi label classifiers for secure semantic similarity join. We compared RBF Networks, SVM, and kNN multi label classifiers. We used a subset of the Pubmed dataset consisting of an 800 records training set, with 10% selective random records to be added later, and a 3000 records testing set. Each record is allowed to have up to four labels. We used k = 3, and we used the polynomial kernel SMO for SVM. Finally, we used the SVM single label classification results as a baseline. Fig. 11 depicts the results. Clearly, using multi label classification outperformed single label classification in terms of F_1 measurement. This is reasonable because the ideal

performance of any single label classifier will not exceed $\frac{1}{N}$ of its corresponding multi label classifier, where N is the maximum allowed number of labels for each record.

Besides, the RBF Network classifier outperformed both SVM and kNN classifiers in this dataset. This is due to its non linear nature, in contrast with the polynomial SVM and lazy kNN. The Pubmed dataset, due to its overlapped topics and some noisy records, needs a non linear classifier to produce the best classification accuracy. One more advantage of an RBF Network classifier is that it is not highly affected by the parameter optimization step. An SMO classifier has an RBF non-linear kernel option, which could be comparable to that of an RBF Networks classifier; however, it performs poorly without the parameter optimization step.

In order to study the effect of adding selective random records and changing epsilon value on the multi-label classification accuracy, we used the RBF Network classifier with various epsilon values ranging from 1 to 100, and various portions of the selective random records ranging from 10% to 50%. Fig. 12 illustrates the findings. We noted that both increasing the portion of the added selective random records and increasing the epsilon value decreased the classifier F_1 measurement. However, as we discussed previously, we do not need to add a large portion of the records nor largely increase the epsilon value. Adding a small portion with a small epsilon value would provide an adequate level of privacy without affecting the F_1 measurement. Furthermore, the added portion of random records and the epsilon value are domain dependant, and depend on the domain error tolerance and the required level of privacy.

Figure 12. The effect of changing epsilon and and adding selective random records proportional to the dataset size on the multi-label classification. RBF classifier was used. Obviously, increasing the added selective random records and increasing epsilon decreases F_1 measurement.



7 Dynamically Expandable Secure Semantic Similarity Join Protocol For Long Attributes Using Supervised Learning

The classification categories are not always static. Commonly, new categories could be created over time. Our protocol should have the ability to expand to include such new categories. There are many real life applications that need such expansion. Hereafter, we list two examples.

Example 7.1: New Diseases Detection
Recently, new diseases have been brought to the world's attention. The ability to detect new diseases is crucial. The existing protocols should be able to detect when test cases that belong to new non existing labels are being introduced. Such ability can speed the detection process and minimizes its consequences. Moreover, the retraining process is also important to consider the newly added categories when classifying new test cases.

Example 7.2: Dividing Movie Classifications
In many cases, one starts with an initial number of categories, and later, one category is divided into two categories or more. For example, in the past, movie categories were limited. However, over time, each category started to contain many subcategories, and the differences among these subcategories have been increased. This process is a continuous process, and the existing protocols are supposed to detect when the category needs to be divided, and to retrain itself on the new subcategories.

The basic model for such an expandable supervised protocol is in the following.

Protocol4: Expandable Secure Protocol for Semantic Similarity Join using Long Attributes for Supervised Learning.
Input: A new test case arriving a source.
Output: Determine if there is a need to divide a category or introduce a new one.
Protocol:

(1) The source classifies the test case using the training model, as proposed in Protocol2, and join it to the shared records of its same label.

(2) The source updates a shared flag, which is used to detect the confidence of the assignment and the state of the category after the assignment.

(3) If the flag exceeds a defined threshold, divide that corresponding cluster into two clusters using a clustering protocol, change the labels assigned to the records in that divided cluster to reflect the new clusters.

(4) Retrain the Classifier using the new labels, and share the updated training model among the sources.

As the source will be granted an access to the records of the same label as its input test case, the source would be able to apply the measurement without violation in privacy. The measurement used to detect classification confidence could be silhouette measurement, sum of squared distance from center, and so on. Later, a suitable retraining process is applied.

The two factors that play key role in the expandable secure protocol are the detection method and the retraining method. Therefore, it is worthwhile to study them in depth. It should be noted that before expanding the existing categories, every record is represented using the existing categories as dimensions. This representation is done using the Chi square feature selection in line 03 of Protocol 2. In this record representation, all the existing categories are supposed to be included equally by having the same number of dimensions. When we expand the categories, the new category must be included, along with the existing ones, in the representation of every new record. This leads to a fair classification in the later steps. Merely dividing the cluster into two clusters and changing the labels of their records would not achieve such fair classification because the new category does not have its "quota" in the representation of the records. The expandable secure protocols topic needs much more work, and we will leave it for future studies.

8 Conclusions

In this work, we propose efficient secure protocols for semantic similarity join using long string join attributes. We consider both the supervised and the unsupervised cases. For the unsupervised case, we show that the diffusion maps method provides the best performance, when compared with other semantic similarity methods, in joining long strings. Both mapping into the diffusion map space and adding a small portion of randomly generated records can hide the original data without affecting accuracy.

For the supervised case, we propose an efficient secure protocol for long string join attributes that uses Diffusion Maps and selective random records, which are hard to detect and does not affect the classification accuracy. Moreover, we study its performance on the multi-label classification scenario, when every record can refer to multiple entities simultaneously. Finally, we introduce the expandable supervised protocol and provided a basic model that could be the basis for future work. Studying the effect of using selective random records instead of pure random records in the secure unsupervised protocol is another future work direction.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions, which contributed in further improving the quality of the paper.

References

[1] Agrawal, R., Evfimievski, A., and Srikant, R. (2003) Information Sharing Across Private Databases. In *Proceedings of SIGMOD* 2003, pages 86-97. ACM

[2] Freedman, M.J., Nissim, K., and Pinkas, B. (2004) Efficient Private Matching and Set Intersection. In *Proceedings of EUROCRYPT* 2004, pages 1-19.

[3] Scannapieco, M., Figotin, I., Bertino, E., and Elmagarmid, A. K. (2007) Privacy Preserving Schema and Data Matching. In *Proceedings of SIGMOD* 2007, pages 653-664.

[4] Hawashin, B., Fotouhi, F., and Grosky, W. (2010) Diffusion Maps: A Superior Semantic Method to Improve Similarity Join Performance. In *Proceedings of the International Conference of Data Mining Workshops (ICDM Workshops* 2010), pages 9-16. IEEE

[5] Karakasidis, A. and Verykios, V.S. (2009) Privacy Preserving Record Linkage Using Phonetic Codes. In *Proceedings of Balkan Conference in Informatics (BCI* 2009), pages 101-106.

[6] Hjaltason, G.R. and Sarnet, H. (2003) Properties of Embedding Methods for Similarity Searching in Metric Spaces: IEEE TPAMI 25, 5, pages 530-549.

[7] Churces, T. and Christen, P. (2004) Some Methods for Blindfolded Record Linkage: BMC Medical Informatics and Decision Making, 4,9, pages 1-17.

[8] Elmagarmid, A., Panagiotis, G., and Verykios, S. (2007) Duplicate Record Detection: A Survey: IEEE TKDE, 19,1, pages 1-16.

[9] Coifman, R.R. and Lafon, S. (2006) Diffusion Maps: Applied and Computational Harmonic Analysis 12,1, pages 5-30.

[10] Deerwester, S., Dumais, S., Furnas, G., Landauer, T., and Harshman, R. (1990) Indexing by Latent Semantic Analysis: Journal of the American Society for Information Science, 41, 6, pages 391-407.

[11] Ataa-Allah, F., Grosky, W. I., and Aboutajdine, D. (2008) Document Clustering Based on Diffusion Maps and a Comparison of the k-Means Performances in Various Spaces. In *Proceedings of IEEE Symposium on Computers and Communications (ISCC 2008)*, pages 579-584. IEEE

[12] He, X. and Niyogi, P. (2003) Locality Preserving Projections: Advances in Neural Information Processing Systems.

[13] IMDB Website: http://imdb.com.

[14] Amazon Website: http://amazon.com.

[15] http://www.zjucadcg.cn/dengcai/Data/data.html

[16] Tang, B., Shepherd, M., Milios, E., and Heywood, M. (2005) Comparing and Combing Dimension Reduction Techniques For Efficient Test Clustering. In *Proceedings of Society for Industrial and Applied MathematicsWorkshops (SIAM Workshops 2005)*.

[17] Al-Lawati, A., Lee, D., McDaniel, P. (2005) Blocking-aware Private Record Linkage. In *Proceedings of ACM SIGMOD workshops* 2005, pages 59-69. ACM

[18] Chaudhuri , S., Chen, B. C., Ganti , V. Kaushik , R. (2007) Example-Driven Design of Efficient Record Matching Queries. In *Proceedings of the International Conference on Very Large Databases (VLDB 2007)*, pages 23-27.

[19] Minton, S. N., Nanjo, C., Knoblock, C. A., Michalowski, M., Michelson, M. (2005) A Heterogeneous Field Matching Method for Record Linkage. In *Proceedings of the International Conference of Data Mining (ICDM 2005)*, pages 314-321. IEEE

[20] Bilenko, M. and Mooney, R.J. (2003) Adaptive Duplicate Detection Using Learnable String Similarity Measures. In *Proceedings of the Special Interest Group of Knowledge Discovery and Data Mining(SIGKDD 2003)*, pages 39-48. ACM

[21] Kumar, C.A. and Srinivas, S. (2006) Latent Semantic Indexing Using Eigenvalue Analysis for Efficent Information Retreival: Int. Journal of Applied Mathmatics and Computer Science., 16,4, pages 551- 558.

[22] Cohen, W., Ravikumar, P., and Fienberg, S. (2003) A Comparison of String Distance Metrics for Name-Matching Tasks. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2003)*, pages 73-78.

[23] Pubmed Dataset http://www.ncbi.nlm.nih.gov/entrez/query.fcgi or http://gateway.nlm.nih.gov/gw/Cmd.

[24] Joachims, T. (1998) Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proceedings of European Conference. on Machine Learning (ECML 1998)*, pages 137–42.

[25] Yang, Y. and Pedersen, J. O. (1997) A Comparative Study on Feature Selection in Text Categorization. In *Proceedings of the International Conference on Machine Learning (ICML 1997)*, pages 412-420.

[26] Praks, P., Machala, L., and V., Snasel (2007) On SVD-free Latent Semantic Indexing for Iris Recognition of Large Databases: V. A. Petrushin and L. Khan (Eds.) Multimedia Data Mining and Knowledge Discovery, 5, 24, pages 472-486.

Appendix A

The 23 subcategories of MeSH category C 'Diseases'

C01 Bacterial Infections and Mycoses
C02 Virus Diseases
C03 Parasitic Diseases
C04 Neoplasms
C05 Musculoskeletal Diseases
C06 Digestive System Diseases
C07 Stomatognathic Diseases
C08 Respiratory Tract Diseases
C09 Otorhinolaryngologic Diseases
C10 Nervous System Diseases
C11 Eye Diseases
C12 Urologic and Male Genital Diseases
C13 Female Genital Diseases and Pregnancy Complications
C14 Cardiovascular Diseases
C15 Hemic and Lymphatic Diseases
C16 Neonatal Diseases and Abnormalities
C17 Skin and Connective Tissue Diseases
C18 Nutritional and Metabolic Diseases
C19 Endocrine Diseases
C20 Immunologic Diseases
C21 Disorders of Environmental Origin
C22 Animal Diseases
C23 Pathological Conditions, Signs and Symptoms