EFFICIENT SIMILARITY JOIN METHOD USING UNSUPERVISED LEARNING

Bilal Hawashin[1], Farshad Fotouhi[2], and William Grosky[3]

[1]Department of Computer Information Systems, Alzaytoonah University of Jordan, Amman 11733, Jordan
hawashin@zuj.edu.jo
[2]Department of Computer Science, Wayne State University, Detroit, MI 48202, USA
fotouhi@wayne.edu
[3]Department of Computer and Information Science, University of Michigan-Dearborn, Dearborn, MI 48128, USA
wgrosky@umich.edu

ABSTRACT

This paper proposes an efficient similarity join method using unsupervised learning, when no labeled data is available. In our previous work, we showed that the performance of similarity join could improve when long string attributes, such as paper abstracts, movie summaries, product descriptions, and user feedback, are used under supervised learning, where a training set exists. In this work, we adopt using long string attributes during the similarity join under unsupervised learning. Along with its importance when no labeled data exists, unsupervised learning is used when no labeled data is available, it acts also as a quick preprocessing method for huge datasets. Here, we show that using long attributes during the unsupervised learning can further enhance the performance. Moreover, we provide an efficient dynamically expandable algorithm for databases with frequent transactions.

KEYWORDS

Similarity Join, Unsupervised Learning, Diffusion Maps, Databases, Machine Learning.

1. INTRODUCTION

Similarity join is grouping pairs of records whose similarity is within a threshold T. Although many supervised learning methods have been proposed to perform similarity join, when a training set of similar records already exists, in many other real-life cases, it is very expensive or even impossible to create a training set to assist in the similarity join method. In this case, a similarity join method could be constructed using unsupervised learning techniques. Moreover, when the dataset is huge, unsupervised methods could be used as a quick preprocessing method to group candidates of similar records together. These methods can filter the dataset by eliminating most of the non similar records in a timely manner. Later, accurate but more expensive methods could be used to detect the similar records. Many methods have been proposed to solve the unsupervised similarity join problem [12][13], however, to our knowledge, all these solutions have been used mainly with short attributes. 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 contrast, *short string* refers to the data type representing any string value of limited length, such as person name, paper title, and address. *Long attribute* refers to any attribute of long string data type, while *short attribute* refers to any attribute of short string data type.

Example 1: The Use of Similarity Join

In one scenario, we could have two items(records) that belong to the same entity but are written differently in the two sources. This could be due to spelling errors, abbreviations, synonyms, different formatting styles in each source, and so on. Our goal is to detect these items and solve the differences in order to perform data integration. For example, two records that refer to the same person in two different tables can have a spelling error in the person name or different date conventions as depicted in Fig. 1. In other scenario, the two items exist in the same source, and our goal here is to detect them to eliminate duplication. In a third scenario, we already assume that the two items belong to different entities and our goal is to group similar items together. For example, in a database of research papers, every paper is different from the others in the database. Our goal here is to group similar papers together according to their content.

Human Resources Section

Person First Name	Person Last Name	Date of Birth	...
John	Smith	1/6/1980	

Finance Section

Person First Name	Person Last Name	Date of Birth	...
Jonh	Smith	Jun - 1 - 80	

Figure 1. Similarity Join Motivation.

Example 2: The Use of Unsupervised Learning

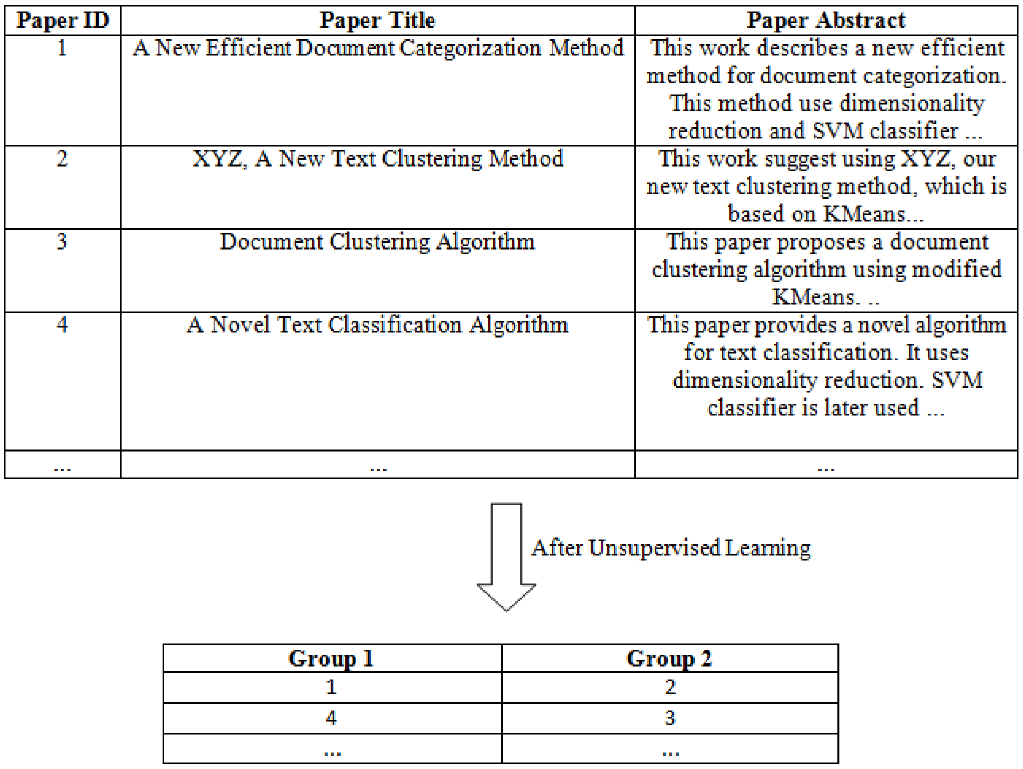
Assume that we have a huge dataset of medical research papers with attributes such as Paper Title, Paper Authors, and Paper Abstract. Our objective is to group similar papers together according to their content. Assume also that we do not know already any similar papers. In this case, unsupervised methods are needed to perform this objective.

Each output cluster would represent candidate similar papers. Later, an accurate but expensive algorithm could be used to detect real similar papers within each cluster. Therefore, unsupervised methods can save time by minimizing the number of comparisons done by the second expensive algorithm. Also, using unsupervised methods is one main solution when no training set of similar papers exists. Fig. 2 depicts the use of unsupervised learning.

We showed in our previous work [9][15] that using long attributes instead of short attributes in the supervised learning would improve the similarity join performance. Therefore, it is worthwhile to study the use of long attributes in unsupervised similarity joins.

Example 3: The Use of Long Attributes

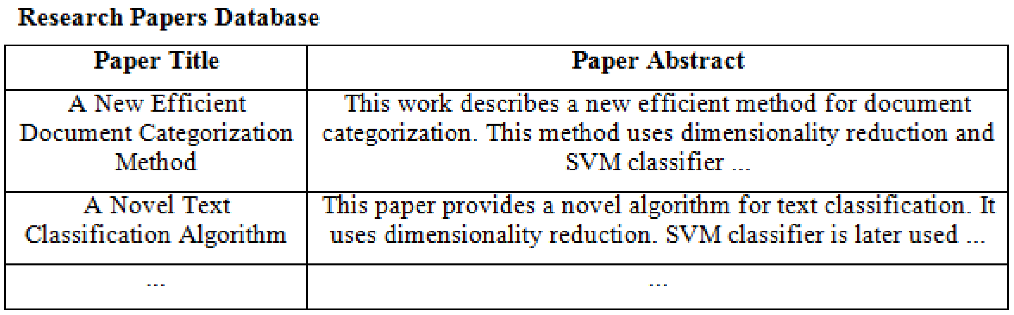
In our previous dataset, instead of using short attributes, such as Paper Title, to detect similar papers, we want to study the use of long attributes, such as Paper Abstract. Fig. 3 depicts an example. Obviously, long attributes are longer and contain more information than short attributes. Therefore, we need to study the effect of using long attributes instead of short ones. We find the pairwise similarities of these long values using a suitable similarity measurement that will be decided later. This similarity method needs to consider the semantic similarities among long values as illustrated in Fig. 3. Next, we cluster the resulting pairwise similarity matrix using unsupervised methods. The output clusters represent the similar papers. Therefore, our first objective is to find a suitable similarity measurement for long attributes. Later, we compare the effect of using long attributes versus short attributes on the unsupervised similarity join performance.



Paper ID	Paper Title	Paper Abstract
1	A New Efficient Document Categorization Method	This work describes a new efficient method for document categorization. This method use dimensionality reduction and SVM classifier ...
2	XYZ, A New Text Clustering Method	This work suggest using XYZ, our new text clustering method, which is based on KMeans...
3	Document Clustering Algorithm	This paper proposes a document clustering algorithm using modified KMeans. ..
4	A Novel Text Classification Algorithm	This paper provides a novel algorithm for text classification. It uses dimensionality reduction. SVM classifier is later used ...
...	...	...

Group 1	Group 2
1	2
4	3
...	...

Figure 2. The Use of Unsupervised Learning.



Research Papers Database

Paper Title	Paper Abstract
A New Efficient Document Categorization Method	This work describes a new efficient method for document categorization. This method uses dimensionality reduction and SVM classifier ...
A Novel Text Classification Algorithm	This paper provides a novel algorithm for text classification. It uses dimensionality reduction. SVM classifier is later used ...
...	...

Figure 3. An Example of a Short Attribute (Paper Title) and a Long Attribute (Paper Abstract).

On the other hand, databases are intrinsically dynamic. Records are inserted, updated, and deleted frequently. This could change the number of clusters produced by the unsupervised methods accordingly. Most of the previous work assumed the database static. Therefore, our second objective is to provide a similarity join method under unsupervised learning that is dynamically expandable with continuous transactions.

This work is divided into four phases. First, finding the best semantic method for joining long attributes using unsupervised learning techniques. Second, comparing the effect on performance when joining long attributes versus joining short attributes using unsupervised learning. Third, providing and evaluating our similarity join unsupervised method. Fourth, providing a solution that is efficient for databases with frequent transactions. It should be noted that the comparison with the previous work is done in phase two, as there is no previous work to be compared with in phase one. Besides, many short string methods were not included in this comparison of phase one because of their high running-time cost and low accuracy when applied to long string values. Fig. 4 summarises the phases of our work.

In phase one, we compare diffusion maps [1], latent semantic indexing(LSI) [2], eigenvectors [3], and independent component analysis(ICA) [16]. In phase two, we compare the best method from phase one with similarity methods for short attributes from the literature such as TF.IDF and SoftTF.IDF [5]. KMeans [17] was used to cluster the output of each method. In order to evaluate the performance, we used three datasets: Amazon Product Descriptions [10], IMDB Movies dataset [6], and PubMed [8].

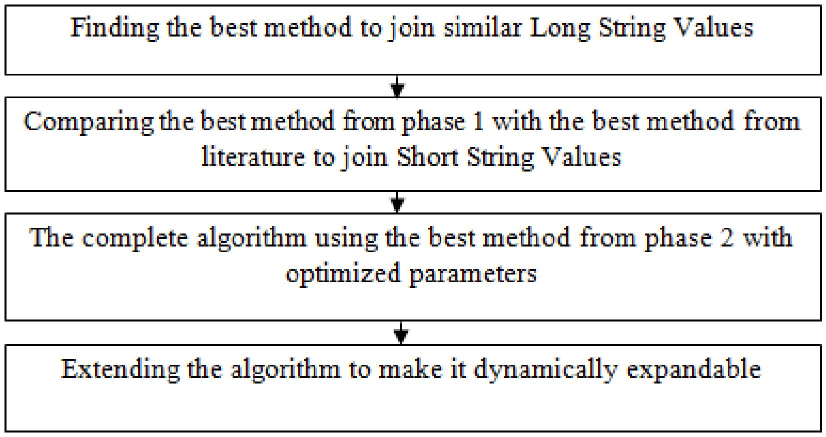

Finding the best method to join similar Long String Values
Comparing the best method from phase 1 with the best method from literature to join Short String Values
The complete algorithm using the best method from phase 2 with optimized parameters
Extending the algorithm to make it dynamically expandable

Figure 4. The Phases of Our Work

The contributions of this work are as follows.

- Adopting the use of long attributes to replace or assist the short attributes in order to enhance the similarity join performance under unsupervised learning techniques.
- Finding an efficient semantic method that can be used for joining values of long attributes.
- Providing a dynamically expandable algorithm for databases with frequent transactions.

The rest of this paper is organized as follows. Section 2 describes both the candidate semantic methods and the used datasets. Besides, it compares these semantic methods for joining long attributes using unsupervised learning. Section 3 compares the performance upon using long attributes against using short attributes. Section 4 explains our proposed similarity join method for long attributes using unsupervised learning. Section 5 introduces the databases with frequent transactions scenario and provides a solution for such an issue. Finally, Section 6 is our conclusions.

2. COMPARING SEMANTIC SIMILARITY JOIN METHODS FOR LONG ATTRIBUTES USING UNSUPERVISED LEARNING

In this section, we describe the candidate semantic methods and the used datasets. Later, we compare these semantic methods for joining long string attributes when unsupervised learning is used. The best method is used as part of our Algorithm later. We compare diffusion maps, LSI, eigenvectors, and ICA. Both LSI and diffusion maps use singular value decomposition(SVD) to eliminate noise and emphasise the semantic relationships, however the input matrices are different. In diffusion maps, the input is a long value by long value similarity square matrix, while in LSI, the input is term by long value matrix. Eigenvectors is an approximation of LSI without using the expensive SVD operation, and ICA finds the main projections in the data. The output matrix of each method is a long value by reduced number of dimensions matrix.

2.1 Candidate Semantic Methods

The detailed description of these method is as follows.

2.1.1 Diffusion Maps

Diffusion maps is a dimensionality reduction method proposed by Lafon [1]. 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, 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, 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 *C* of N long string values, represented as vectors. Let *x*,*y* be any two vectors in *C*, $1\leq i,j\leq N$. A weighted matrix $W_\sigma(x,y)$ can be constructed as

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

where σ specifies the size of the neighborhoods that defines the local data geometry.

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

We can create a new kernel as follows:

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

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

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

Suppose $d_\sigma(x) = \sum_{y \varepsilon C} W_\sigma^\alpha(x,y)$, (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 C. The diffusion distance D_t between *x* and *y* at time *t* of the random walk is

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

where ϕ_0 is the stationary distribution of the Markov chain.

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

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

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

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

We used the values of σ and α to be 10 and 1 respectively for experiments as used in the literature.

2.1.2 Latent Semantic Indexing

Latent Semantic Indexing [2] uses the Singular Value Decomposition operation to decompose the term long string value matrix *M*, that contains terms 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 present in the dataset. When the components are truncated to *z* dimensions, a reduced representation matrix, M_z is formed as

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

2.1.3 EigenVectors

Here, the eigenvectors and their corresponding eigenvalues are extracted directly from the term long string value matrix [3]. Originally, each long string value is represented as a combination of all eigenvectors and their eigenvalues. A reduced number of eigenvectors, with their corresponding eigenvalues, is selected that captures most of the dataset information.

2.1.4 Independent Component Analysis

Independant Component Analysis[16] is a used for revealing hidden factors that underlie sets of random variables. ICA finds a model for the observed multivariate dataset. In the model, the data variables are assumed to be linear mixtures of some unknown latent variables, and the mixing system is also unknown. The hidden variables are assumed non-gaussian and mutually independent, and they are called the independent components of the observed data. These independent components, also called sources or factors, can be found by ICA. A reduced number of latent variables can be used instead to eliminate the noise and capture the semantic relations in the dataset.

For Diffusion Maps, we used Lafon Matlab implementation[1]. We used the values of σ and α to be 10 and 1 respectively as used in the literature. Regarding LSI, we used the SVDs( ) Matlab built-in function. For the eigenvectors method, we used the Eigs( ) Matlab function. For ICA, we used FastICA package [18].

We use only dimensionality reduction methods as candidate semantic methods because the clustering process is very sensitive to the number of dimensions. Using non-dimensionality reduction methods such as TF.IDF with cosine similarity as input to unsupervised methods will increase significantly the clustering time because of the relatively large number of dimensions. In order to evaluate the previous methods in joining long string values, two datasets are used, which are Amazon products and IMDB. It should be noted that the number of records in the datasets is irrelevant to the performance of the algorithms, as records are processed sequentially. In other words, all the algorithms, except Algorithm 1, are repeated for each record. Therefore, the data size for each run is always one record, which makes the actual number of records irrelevant to the performance. Regarding Algorithm 1, it uses a fixed and relatively small initial number of records, and it does not need a large number of records to proceed.

For our experiments, 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, and Matlab 2008a for the implementations of the candidate semantic methods and KMeans.

In this phase, for the Movie Summary attribute in IMDB Dataset, the stopwords were removed using a list of 429 words [19], and the text was converted into lowercase. The term-long string value frequency matrix was then generated, where every row in this matrix represents a word, and every column represents a long string value. Later, the TF.IDF [7] weighting matrix was computed. TF.IDF is a commonly used weighting method in the text mining literature. The TF.IDF weighting of a term *w* appearing in a long string value *x* is given as follows:

$$\mathrm{Tf.Idf}(w,x)=\log(\mathrm{tf}_{w,x}+1).\log(\mathrm{idf}_{w}), \qquad (11)$$

where $\mathrm{tf}_{w,x}$ is the frequency of the term *w* in the long string value *x*, idf_w is $\frac{N}{n_w}$, where *N* is the number of long string values in the database *C*, and n_w is the number of long string values in the database that contains the term *w* in their corresponding attribute.

Next, we used Mean TF.IDF unsupervised dimensionality reduction method [11] to eliminate insignificant words, and we selected the 2% of the features with the highest importance. The values in the Product Description attribute from Amazon Products datasets were processed similarily.

The performance measurements used in this phase were *silhouette value, purity, clustering time, and operation* time. They are defined as follows:

Silhouette value for a record *x*, which is assigned to a cluster *c* of *n* records, is a measurement of the assignment suitability for this point during the clustering process. It is calculated using the following equation.

$$Silh(x)=\frac{b(x)-a(x)}{\max(b(x),a(x))}, \qquad (12)$$

where $b(x)$ is the average distance between *x* and all the records in the same cluster, and $a(x)$ is the average distance between *x* and all the records in other clusters.

Purity measures the overall clustering accuracy in correspondence with the actual cluster labels[14]. Let $C = \{C_1, C_2, C_3, \ldots, C_k\}$ represents the set of clusters, and let $L = \{L_1, L_2, L_3, \ldots, L_m\}$ represents the set of labels (classes). Purity is calculated using the following equation.

$$\text{Purity}(C,L) = \frac{\sum_k \max_m (C_k \cap L_m)}{n}, \qquad (13)$$

Where *n* is the total number of points in the dataset. Both Purity and clustering accuracy are used alternatively in this context.

Clustering time is the time required to perform the clustering algorithm.
Operation time is the time required to perform the dimensionality reduction operation on the dataset.

2.2 Data Description

The following is a brief description of the used datasets.

2.2.1 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. The average number of words in each product description is 72 words. An example of Amazon Product Description is the following.

"The Sony DSC-HX200V Cyber-shot Digital Camera with images in 18MP and fluid full HD videos in 1080p using its 1/2.3in CMOS Sensor and 30x optical zoom."

2.2.2 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. The number of words in each summary is 115 words.

2.2.3 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 In our experiments, we used 1104 records that belong to 23 categories. As a long string attribute, we used the paper abstract. The average number of words in each Abstract is 172 words. PUBMED citations and abstracts could be accessed by PUBMED via . http://www.ncbi.nlm.nih.gov/pubmed/.

2.3 Evaluation

After applying each of the semantic methods on the datasets, the KMeans clustering algorithm was used. We used KMeans, which is an example of a partitional clustering method, because it outperformed both hierarchical and suffix tree clustering methods[14]. Furthermore, Nearest Neighborhood methods compare records in a window of size *x*, which does not produce the optimal result because clusters could be of various sizes practically. During this phase, we experimentally selected the optimal number of reduced dimensions and the optimal number of clusters for KMeans as given in Algorithm 1. We used the highest silhouette value after clustering with KMeans to indicate the optimal number of reduced dimensions and optimal number of clusters. In detail, we used a fixed initial value for the number of clusters, 10 clusters for example, and used KMeans with that value to cluster the output of the diffusion maps algorithm using various numbers of dimensions, 10 to 50 dimensions for example. After finding the optimal number of diffusion maps dimensions, we used it with KMeans clustering with various number of clusters. Fig. 5 displays this step. The other semantic methods were manipulated similarly. Later, we used both clustering time and cluster purity to evaluate the accuracy of the resulting clusters. The comparison of the semantic methods according to the clustering time for Amazon and IMDB datasets showed no significant differences because the number of reduced dimensions is the same. Fig. 6 and Fig. 7 show the comparison of the four methods according to the purity in Amazon and IMDB datasets respectively. In Fig. 7, when the methods were applied to the IMDB dataset, which is relatively easy and contains disjoint clusters, the four methods showed high performance, and the performance decreased in LSI, ICA, and eigenvectors when using the Amazon dataset, which is more complex and contains overlapped clusters. Diffusion maps proved to have the most stable performance. It can handle overlapped clustering that has higher error rate. Table 1 shows the operation times, where the methods were ordered as LSI < EIG < Diff < ICA.

This is due to the larger amount of information contained in the input matrix of the ICA and diffusion maps, which are document-by-document matrices, in contrast with the simple, relatively sparse input matrices to LSI and eigenvectors. Diffusion maps operation time is not very small, in contrast with ICA, and could be compensated with the gain in accuracy upon using this method. As diffusion maps showed the best performance, it was adopted in our solution.

Table 1. Operation Time (In Seconds) Of The Candidate Methods For Joining Long Attributes In The Two Datasets

Method	IMDB	Amazon
Diffusion Maps	2.5	1.35
LSI	0.24	0.1
ICA	10	3.6
Eigenvectors	0.45	0.23

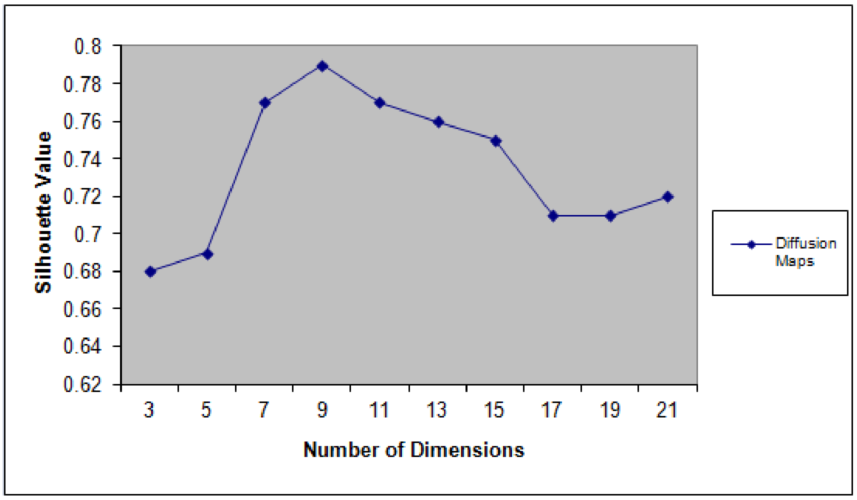


Figure 5. Determining the best number of clusters for KMeans under diffusion maps space. The best number of dimensions was nine dimensions. We used 700 product descriptions from Amazon Products dataset.

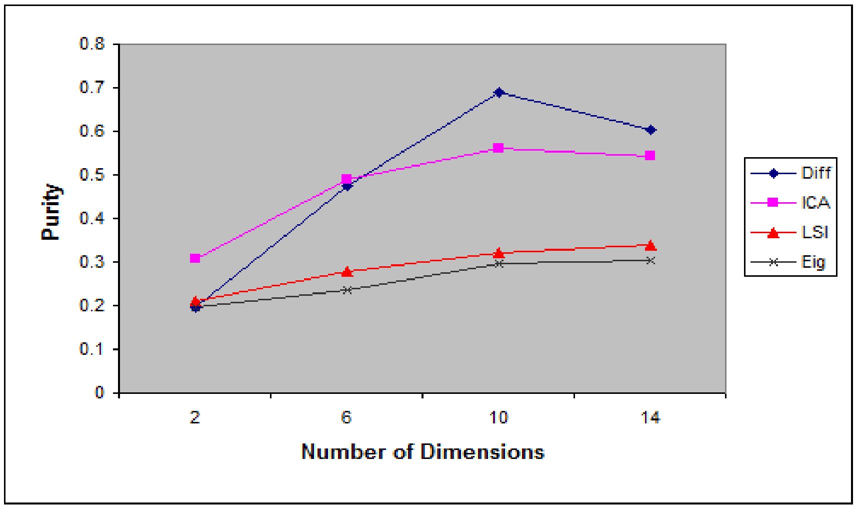


Figure 6. Comparing the purity of the KMeans clustering under diffusion Maps, ICA, LSI, and eigenvectors. Diffusion Maps showed the best performance. We used 700 product descriptions from Amazon Products dataset.

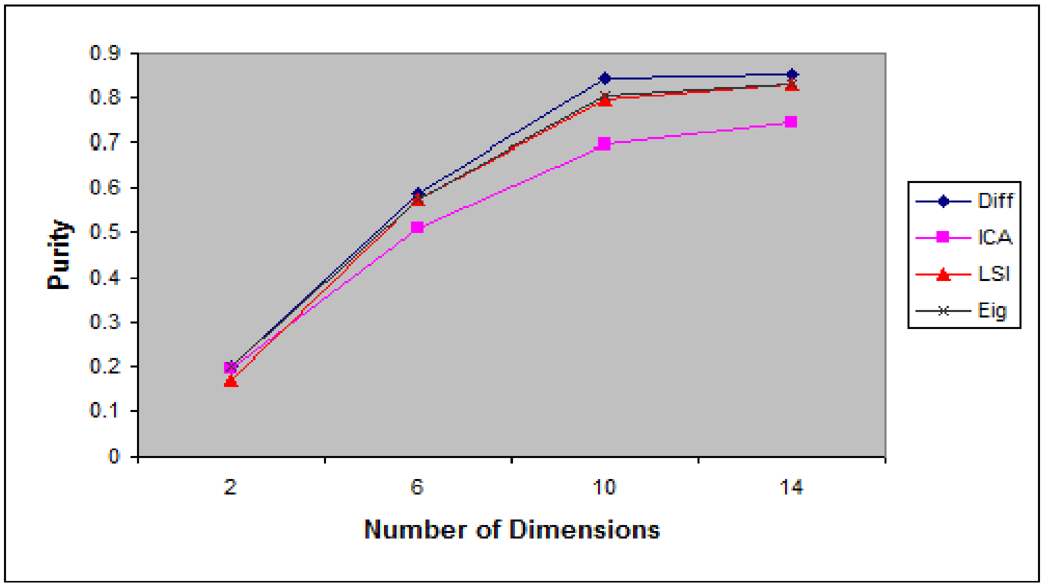


Figure 7. Comparing the purity of the KMeans clustering under diffusion Maps, ICA, LSI, and eigenvectors. Diffusion Maps showed the best performance. We used 1000 movie summaries from IMDB dataset.

3. LONG STRING VS SHORT STRING EVALUATION

For phase two, we compared the best semantic method for long attributes with top existing methods for short attributes. According to phase one, diffusion maps proved to be the best semantic method, among the compared ones, for long attributes, when no training set exists. In this phase, clustering using long attributes represented in diffusion maps space was compared with clustering using short attributes represented using existing short attribute methods.

We used Product Title and Product Description attributes from Amazon products dataset to represent the short attributes and the long attributes, respectively. We used 700 records for this purpose. For long attributes, the Product Description values were represented in diffusion maps space. For short attributes, Product Title values were represented using pairwise SoftTF.IDF [4] similarities, pairwise SoftTF.IDF similarities reduced using diffusion maps, and pairwise TF.IDF similarities reduced using diffusion maps. The detailed description of SoftTF.IDF is given as follows. The SoftTfIdf similarity between two string values X and Y is given as follows:

$$SoftTfIdf(X,Y) = \sum_{w \in CLOSE(\theta,X,Y)} V(w,X)\,V(w,Y)\,D(w,Y), \qquad (14)$$

Whereas $V(w,X)$ represents the Tf.Idf weighting of the term w in the string value X, $V(w,Y)$ represents the Tf.Idf weighting of the term w in the string value Y, and $\mathrm{CLOSE}(\theta, X, Y)$ represents all terms $w \in X$ such that there is some term $v \in Y$ such that $D'(w,v) > \theta$. $D(w,v)$ denotes Jaro-Winkler distance between the terms w and v.

$D(w,Y) = \max_{v \in Y} (D(w,v))$. For our experiments, we used $\theta = 0.9$ as adopted in the literature.

This method is a superior method for joining short string values [4], and therefore, it is worthwhile to study its performance on long string values.

After applying the previous methods, KMeans was used to cluster the output of these methods separately. It should be noted that we did not use many existing unsupervised similarity join methods such as [12][13] because of their poor performance with long string values. We used two performance measurements, purity and clustering time. Table 2 depicts the results.

Clearly, KMeans clustering of long string values represented by diffusion maps proved to have the best purity, which is reasonable because long attributes tend to have much more information than short attributes, which will increase the clustering accuracy. Using the clustering time, all the previous methods were comparable except the SoftTF.IDF alone because it is not a dimensionality reduction method, and the number of dimensions affects significantly the clustering time performance. Overall, we conclude that using diffusion maps semantic method with long attributes showed a better performance than using the existing unsupervised similarity join methods that use short attributes.

Table 2. Kmeans Clustering For Long And Short Attributes. Clustering Time Is Measured In Seconds

Method	**Purity**	**Clus_T**
Prod Desc (Diff)	0.69	0.05
Prod Title (SoftTF.IDF)	0.405	1.2
Prod Title (SoftTF.IDF+ Diff)	0.41	0.08
Prod Title (TF.IDF + Diff)	0.51	0.1

4. SIMILARITY JOIN METHOD FOR LONG ATTRIBUTES USING UNSUPERVISED LEARNING

After showing that using long string attributes with diffusion maps and clustering the output using KMeans can provide an efficient performance in comparison with other unsupervised similarity join methods, we adopt this in our algorithms. In this section, we provide and discuss our unsupervised similarity join method, and evaluate its performance on new testing records. Basically, our method is composed of two algorithms, Algorithm 1 and Algorithm 2. Algorithm 1 takes as an input an initial set of unlabelled records and applies the similarity join operation on them using long attributes and diffusion maps. The output of this algorithm is a set of clusters, where every cluster represents a set of records that are joined according to their semantic similarity. Algorithm 2 takes as an input the set of clusters from Algorithm 1, and for every newly arriving testing record, it will assign it to one of the existing clusters. We explain the details of each algorithm next.

In Algorithm 1, the input is a dataset represented as a term-document matrix, where each record represents a term (word) and every column represents a long string value. The output is a set of clusters, where every cluster represents a set of semantically similar items.

In Algorithm 1, after preprocessing the dataset by applying the TF.IDF weighting and reducing the dimensionality using the Mean TF.IDF unsupervised dimensionality reduction method, the diffusion maps method is applied to obtain the reduced representations of the long string values, Y, as stated in line 11. Every output row in Y represents a long string value, and every column represents a reduced dimension. Later, the KMeans algorithm is applied to cluster the long string values in the reduced space, and the silhouette value is calculated. We need to select the optimal values of both the number of dimensions *Z* in line 11 and *Num_Clusters* in line 27 experimentally in order to maximize the silhouette value. After obtaining the optimal Z and *Num_Clusters*, KMeans is applied using both values to output the optimal set of clusters. It should be noted that this algorithm is applied once only, and it is applied to any initial set records.

After obtaining the set of clusters using Algorithm 1, Algorithm 2 is an incremental algorithm that is used to assign every newly arriving record to its suitable cluster among the existing clusters. Algorithm 2 converts the arriving testing record into its reduced diffusion maps representation, as in line 1. Next, in line 4, it finds the cosine similarity between the reduced testing record representation and all the cluster centroids. In line 8, the testing record is assigned to the cluster whose centroid is the closest.

In the evaluation part, Algorithm 1 was already evaluated in the previous section and it outperformed the compared unsupervised similarity join methods. In order to evaluate Algorithm 2, we inserted various numbers of records belonging to existing clusters, and we computed the similarity join accuracy, which represents the record-cluster assignment accuracy. Three datasets were used in this experiment, which are IMDB[6], Amazon Products[10], and PubMed[8], and the results are illustrated in Table 3.

Clearly, the algorithm can assign the newly arriving records to the existing clusters with a high accuracy. It is obvious also that its accuracy is data-dependant. The algorithm works better with datasets of disjoined clusters, such as IMDB, than those of overlapped ones, such as Amazon Products.

Table 3. Algorithm 2 Accuracy on the Three Datasets

Method	**Avg. Accuracy**
IMDB	0.89
PubMed	0.76
Amazon Products	0.73

Algorithm 1 : INITIAL SIMILARITY JOIN METHOD USING LONG STRING VALUES

Input: The term by long string value matrix *M* for the set of unlabeled *D* records

Output: Candidate similar records *Y_Clustered_Opt* represented as clusters.

Algorithm

```
Apply TF.IDF and MeanTF.IDF on matrix M.

Find the pairwise cosine distances among the long string values in M and store it in Dcos.

Select a fixed value for Num_Clusters.

Make multiple runs with different number of dimensions Z in each run.

    Apply Diffusion Maps using Dcos as kernel and Z dimensions. The output of every run will be Y, which represents long string values in the Diffusion Maps reduced space.

    Cluster Y using KMeans and the already fixed Num_Clusters to produce Y_Clustered.

    Find the Silhouette for the output clusters.

    The optimal number of dimensions, Z_Opt will be the one that results in the higest silhouette value.

    Now, apply Diffusion Maps using Z_Opt to get Y_Opt.

    Make multiple runs and use different Num_Clusters in each run.

    Apply KMeans using Y_Opt and Num_Clusters.

    Find the silhouette for the output clusters.

    Use Num_Clusters_Opt corresponding to largest silhouette value.

    Apply KMeans using both Z_Opt and Num_Clusters_Opt to get the optimal clusters. Y_Clustered_Opt.

Return Y_Clustered_Opt
```

```
Algorithm 2:   SIMILARITY JOIN METHOD USING
               UNSUPERVISED LEARNING
Input:     A new testing record t arriving a source.
Output:  Add the testing record to a cluster from Algorithm 1
Algorithm:
(1)  Convert the new test record t into the Diffusion Maps
(2)  reduced representation t_red.
(3)
(4)  Find the cosine similarity between t_red and every
(5)  cluster centroid. Clusters are already produced from
(6)  Algorithm 1.
(7)
(8)  Add the testing record t to the cluster with max
(9)  Cos_Sim[i].
```

5. DYNAMICALLY EXPANDABLE SIMILARITY JOIN ALGORITHM USING LONG ATTRIBUTES

The clustering categories are not always static. Commonly, new categories could be created over time. Our method should have the ability to expand to include such new categories. There are many real life applications that need such expansion. One example is detecting when patients of new disease come for diagnostic, our method needs to detect that these cases belong to new disease. Another example is when movies of a specific category needs to be divided into subcategories like "Horror" and "Thriller". You can see [15] for detailed explanation.

In the following two subsections, we compare various methods to detect records of non-existing clusters and study the effect of reclustering.

5.1. Detecting Records of Non Existing Clusters

Here, our goal is to detect when records of new non-existing clusters are being introduced. We denote existing-cluster and new-cluster groups with EC and NC respectively. We compare two detection measurement: Cosine Distance, which is the complement of the cosine similarity, and Silhouette value, which is computed using equation 12. These two measurements are computed for every arriving record. The maximum value of the detection measurement is returned, as there will be a value for each cluster . If that value is less than a predefined threshold, we consider the record a NC record. We denote this record a satisfying record, as it satisfied the new-cluster criteria.

In order to compare the two measurements, we used both records of EC and records of NC and computed their detection measurement values using both methods. Clearly, the efficient detection measurement is supposed to distinguish records of EC and records of NC by showing a significant difference between their average measurement values. We used IMDB, PubMed, and Amazon Products datasets. It should be noted that in this scenario, the arriving records are processed sequentially, which makes the dataset size irrelevant to the performance. In other words, The results of using Silhouette and Cosine Distance measurements are displayed in Table 4 and Table 5 respectively. Apparently, using silhouette measurement resulted in a better isolation between both record types. Another observation is that the drop percentage when a NC record is introduced is dataset dependent, as not all datasets have the same properties.

Table 4. Average Silhouette For Existing-Cluster Records And New-Cluster Records.

	EC	NC	Drop
IMDB	0.86	0.57	34%
PubMed	0.81	0.7	14%
Amazon	0.77	0.67	13%

Table 5. Average Cosine Distance For Existing-Cluster Records And New-Cluster Records.

	EC	NC	Drop
IMDB	0.95	0.77	19%
PubMed	0.91	0.82	10%
Amazon	0.91	0.88	3%

5.2 Reclustering Analysis

Reclustering is needed when the number of records belonging to a NC becomes large. Reclustering would create a new cluster(s) to minimize the clustering error. When a criteria reaches a user-defined threshold, reclustering is applied. The criteria could be the number of records with detection measurement less than a specific value. For example, if the number of inserted records with silhouette value less than 0.5 exceeds 50, reclustering is needed, as Algorithm 3 states. Various domains could use various thresholds for silhouette measurement depending on their error tolerance. In order to find a suitable threshold value, we inserted a sample of records that belong to EC, computed the silhouette measurement after each insertion, and found the minimum silhouette value. This value was used as the threshold value. In order to illustrate the motivation behind using a reclustering criteria, we conducted an experiment that calculates the percentage of records with a silhouette value less than the threshold. We used both types of records(EC and NC) separately in two different groups. Two datasets were used here, IMDB and PubMed. The percentage of satisfying records among EC and NC groups was 12% and 69% respectively for IMDB, and it was 27% and 48% respectively for PubMed. It is clear that records of NC clusters have lower silhouette values than those of EC, and that using the minimum silhouette value of the sample as a threshold value is promising. Algorithm 3 provides the expandable solution.

Next, we studied the cost and effect of the reclustering process. Two methods were proposed here: labeling the new records manually, or using a clustering method to label them. Only satisfying records are labeled. Ideally, all the NC records are supposed to satisfy the criteria and none of the EC records are supposed to satisfy it. From Table 6, we can see that around 55% of the NC records satisfied the criteria, and 20% of the EC records did. Regarding the EC records, they would not affect the results as they would be eliminated during the labeling phase.

As labeling records using a clustering method produced more erroneous labels (accuracy 0.66 in contrast with 0.9 for manual labeling), we adopted the manual labeling of records, even though it consumes more time(120 sec. per record in contrast with 100 sec. per dataset using clustering), assuming the average length of the record is 75 words. After labeling, the feature selection method needs to be repeated to include the new cluster(s). Initially, each long string value is represented as a vector of the important terms in the existing clusters. In order to ensure a fair comparison, the important terms from the new cluster needs to be extracted and included in the representation of the long string values. In the IMDB dataset, after inserting 300 new-cluster records of three new clusters into the original dataset, which is composed of 1000 records of 10 classes, the feature selection method took 1013 seconds for 13 classes (In comparison to 809 seconds for the 10 basic classes). PubMed showed similar trend. Finally, to study the effect of the reclustering process in the record-cluster accuracy for those records that belong to the newly created cluster, we inserted 490 records that belong to existing and new

```
Algorithm 3: EXPANDABLE SIMILARITY JOIN USING
             LONG ATTRIBUTES UNDER
             UNSUPERVISED LEARNING.
Input:    A new test case arriving a source.
Output: Determine if new cluster is needed.
Algorithm:
  (1)  Use Algorithm 2 to assign the record to
  (2)  one of the existing clusters.
  (3)
  (4)  Find the silhouette value for that record. If
  (5)  it is less than the threshold value,
  (6)  increase a flag variable by one, and insert
  (7)  this record also in a log file.
  (8)
  (9)  If the flag exceeds a defined threshold,
  (10) perform reclustering for records that exist
  (11) in the log file.
```

clusters after performing the reclustering and considering the new clusters, and the accuracy was 0.75, which is sufficient in many domains.

Finally, as an estimation to the frequency of reclustering, we inserted random records from IMDB, and we used various similarity thresholds and various numbers of satisfied records. We recorded the order of that number of satisfying records among the random records. Table 6 represents the results respectively.

Table 6. Reclustering Frequency Using Various Thresholds And Numbers Of Satisfying Records On IMDB

Threshold	**0.8**			**0.7**			**0.6**		
Number Satisfying Records	25	50	75	25	50	75	25	50	75
Order	50	112	161	105	194	328	169	400	-

This table shows the influence of the two threshold parameters on Algorithm 3. It shows that the frequency of reclustering is domain dependant, and it is affected by the used thresholds.

6. CONCLUSIONS

In this work, we proposed an efficient similarity join method for long attributes using diffusion maps and unsupervised learning. This method can create initial set of semantically joined records, and can join newly arriving records to the suitable cluster according to its similarity. Diffusion maps proved to be the best semantic method, among compared, in joining long attributes under unsupervised learning, and KMeans was used later to cluster the records. Furthermore, we showed that joining tables using their long attributes outperformed joining tables using their short attributes. Diffusion maps was used for joining long attributes, and efficient methods from the literature were used for joining short attributes. Both join accuracy and time were improved upon using long attributes. Furthermore, we proposed a model for similarity join for databases with frequent transactions. In this solution, silhouette measurement outperformed cosine distance in the ability to detect records that belong to non-existing clusters. Also, we proposed using a criteria with a threshold value to specify when the reclustering is needed. Such criteria can differ according to the domain.

Future work can be done to extend the reclustering detection methods and the reclustering process. Our dynamically expandable similarity join algorithm servers as a preliminary effort for more detailed and extended work in the future. Besides, the semantic method for joining long values could be improved by extending the comparison and optimizing the parameters.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their comments and suggestions, which contribute in further enhancing the paper quality.

REFERENCES

[1] Coifman, Ronald & Lafon, Stephan, (2006) “Diffusion Maps”, Applied and Computational Harmonic Analysis, Vol. 12, No. 1, pp 5-30.

[2] Deerwester, Scott, Dumais, Susan, Furnas, George, Landauer, Thomas, and Harshman, Richard, (1990) “Indexing by Latent Semantic Analysis”, Journal of the American Society for Information Science, Vol. 41, No. 6, pp. 391-407.

[3] Kumar, Cherukuri & Srinivas, Suripeddi, (2006) “Latent Semantic Indexing Using Eigenvalue Analysis for Efficent Information Retreival”, Intl. Journal of Applied Mathmatics and Computer Science., Vol. 16, No. 4, pp. 551-558.

[4] Cohen, William, Ravikumar, Pradeep, and Fienberg, Stephen, (2003) “A Comparison of String Distance Metrics for Name-Matching Tasks”, In Proc. of the Intl. Joint Conf. on Artificial Intelligence (IJCAI), pp. 73-78.

[5] Russell, Robert, (1922), Index, U.S. Patent 1,435,663, http://patft.uspto. gov/netahtml/srchnum.htm.

[6] IMDB. Accessed June, 2012. http://www.imdb.com.

[7] Yang, Yiming, (1994) “Expert Network: Effective and Efficient Learning From Human Decisions in Text Categorization and Retrieval”, In Proc. of the Intl Conf on Research and Development in Information Retrieval (ACM SIGIR), pp. 13-22.

[8] PubMed Dataset. Accessed June, 2012. http://www.ncbi.nlm.nih.gov/pubmed/.

[9] Hawashin, Bilal, Fotouhi, Farshad, and Grosky, William, (2010) “Diffusion Maps: A Superior Semantic Method to Improve Similarity Join Performance”, In Proc. of ICDM MMIS Workshop, pp. 9-16.

[10] Amazon Website. Accessed June. 2012. http://amazon.com.

[11] Tang, Bing, Shepherd, Michael, Milios, Evangelous, and Heywood, Malcolm, (2005) “Comparing and Combing Dimension Reduction Techniques For Efficient Test Clustering”, In Proc. of SIAM Workshops.

[12] Bradley, Paul & Fayyad, Usama, (1998) “Refining Initial Points for K-Means Clustering”, In Proc. of Intl. Conf. on Machine Learning(ICML), pp. 91-99.

[13] Battacharya, Indrajit & Getoor, Lise, (2004) “Iterative Record Linkage for Cleaning and Integration”, In Proc. of ACM SIGMOD Workshop on Data Mining and Knowledge Discovery, pp. 11-18.

[14] Yoo, Illhoi & Hu, Xiaohua, (2006) “A Comprehensive Comparison Study of Document Clustering for a Biomedical Digital Library MEDLINE”, In Proc. of ACM/IEEE-CS joint conference on Digital Libraries, pp. 220-229.

[15] Hawashin, Bilal, Fotouhi, Farshad, Trauta, Traian, and Grosky, William, (2012) “Efficent Privacy Preserving Protocol for Similarity Join”, Transactions on Data Privacy, Vol. 5, No. 1, pp. 297-331.

[16] Comon, Pierre, (1994) “Independent Component Analysis: a New Concept?”, Signal Processing, Elsevier Vol. 36, No. 3, pp. 287-314.

[17] Loyd., Stuart, (1957) “Least squares quantization in PCM”, IEEE Trans. on Information Theory, Vol 28, No. 2, pp. 129-137.

[18] Gävert, Hugo, Hurri, Jarmo, Särelä, Jakkoo, and Hyvärinen, Aapo, (1996-2005) "FastICA Software Package for Matlab".
[19] Onix Text Retreival Toolkit. Accessed Sep. 2012. http://www.lextek.com/manuals/onix/stopwords1.html.

Authors

Dr. Bilal Hawashin is currently an Assistant Professor in the Department of Computer Information Systems at Alzaytoonah University of Jordan. He received his Ph.D in Computer Science, College of Engineering, from Wayne State University in 2011. Also, he worked in the Department of Computer Information Systems at Jordan University of Science and Technology from 2003-2007. His current research interests include Similarity Join, Text Mining, Information Retrieval, and Database Cleansing. He has various publications in referred journals and conference proceedings. Dr. Hawashin received his B.S. in Computer Science from The University of Jordan in 2002, and his M.S. in Computer Science from New York Institute of Technology in 2003.

Dr. Farshad Fotouhi is currently Dean of the College of Engineering at Wayne State University. He received his Ph.D. in Computer Science from Michigan State University, College of Engineering in 1988. Dr. Fotouhi joined the faculty of Computer Science at Wayne State University in August 1988, where he served as the Department Chair from 2004-2010. Dr. Fotouhi's current research interests include Biomedical Informatics, Semantic Web and Multimedia Systems. He has published over 180 papers in refereed journals and conference proceedings. His research has been supported by NSF, NIH, National Institute of Drug Abuse, Michigan Life Sciences Corridor, Ford Motor Company and many other industries. Dr. Fotouhi has served as a program committee member of various conferences related to his research interests and is currently a member of the Editorial Board of IEEE Multimedia Magazine, Chair of the Steering Committee of IEEE Transactions on Multimedia, and a member of the Editorial Board of the International Journal of Semantic Web and Information Systems.

Dr. William Grosky is currently Professor and Chair of the Department of Computer and Information Science at the University of Michigan-Dearborn. Before joining UMD in 2001, he was Professor and Chair of the Department of Computer Science at Wayne State University, as well as an Assistant Professor of Information and Computer Science at the Georgia Institute of Technology in Atlanta. His current research interests are in Multimedia Information Systems, Text and Image Mining, and the Semantic Web. Dr. Grosky received his B.S. in Mathematics from MIT in 1965, his M.S. in Applied Mathematics from Brown University in 1968, and his Ph.D. from Yale University in 1971. He has given many short courses in the area of Database Management for local industries and has been invited to lecture on Multimedia Information Systems world wide. Serving also on many Database and Multimedia conference program committees, he was an Editor-in-Chief of IEEE Multimedia, and is currently on the editorial boards of many journals.