Efficient Genetic K-Means in Healthcare Domain

Ahmed Alsayat and Hoda El-Sayed

Meta Path-Based Temporal Link Prediction in Heterogeneous Evolving Networks

Alper Ozcan and Sule Gunduz Oguducu

Prediction Interval of Cumulative Number of Software Faults Using Back Propagation Algorithm

Begum Momotaz and Tadashi Doi

An Empirical Study on Recovering Requirement-to-Code Links

Zhang Yuchen, Wan Chengcheng and Jin Bo
Efficient Genetic K-Means in Healthcare Domain

Ahmed Alsayat, Hoda El-Sayed
Bowie State University, U.S.A.

Abstract
Data mining has become an important task to explore the hidden pattern in the data especially in healthcare domain. Increasing health concerns that are interrelated have led to a wide application of health data mining in most healthcare facilities. This paper studies the application of K-means clustering algorithm as one of the data mining application in the healthcare domain. The proposed K-means is an optimized version where it combines the power of Self organizing Map (SOM), Genetic algorithm along with a novel method called Optimized Cluster Distance (OCD) to enhance the quality of the clusters. Self organizing Map (SOM) technique is applied to calculate the optimal number of clusters without providing any number as an input as in the case of traditional K-means. By applying the algorithm to two different healthcare datasets, the proposed method shows a significant improvement in clustering performance and minimizing within cluster distances. The algorithm is successfully evaluated to provide a better quality of clusters compared to other clustering techniques. The paper also shows the clusters accuracy of the proposed algorithm as well as noticeable relationships and insights among clusters using data visualization techniques.

Keywords: K-Means, Genetic Algorithm, Clustering, Data Mining, Self-Organizing Map, Health care.

1. Introduction
Today, the healthcare domain is considered one of the busiest institutions across the globe. As such, the intensity of medical processes and procedures taking place in this industry enables it to generate a lot of medical-related data just like any other busy industry. Medical data is always crucial in treatment and diagnosis processes. For instance, the data obtained in a health facility could be significant in identifying useful patterns and trends that aid in diagnosis and decision-making [1]. Consequently, clustering techniques such as COBWEB, K-means, and D-streams have been applied to analyze patients past health data in determining their fitness or unfitness as far as their health is concerned [2, 3].

Based on the [3] definition, data mining is the process by which data from different sources are combined to come up with a new pattern from a data collection. The earliest data mining is a recent as the 1990’s and comes as a response to increasing collection of data by computers with the hope of gaining deep insights from the data.

The voluminous nature of health data generated by health monitoring systems such as the media sensors, which generates data in video, audio, text, and image forms, is too complex for processing and analysis by conventional methods [4, 5]. Since the 1990s, data mining techniques have been considered effective in transforming the heterogeneous forms of data into usefully significant information that aids in healthcare decision-making and finding deep insights into the hidden structures of the data [3]. In this regard, this paper focuses on the applications of data mining techniques in the healthcare domain. The paper is particularly interested in the application of K-Means clustering algorithm in mining health-related data that are applied to health care.

In a health application such as a brain monitoring system, the K-means technique is an important data mining method that helps in detecting valuable unknown information from huge data sets [4]. Its application in the healthcare sector helps medical researchers in developing effective and efficient healthcare policies, developing patients individual profiles, and creating proper drug recommendation systems [3]. Ideally, the K-means is a clustering technique rather than a classification technique [6, 7]. The implication of this is that it is an unsupervised learning application that has no predefined classes like in the case of the classification techniques [3].

As such, the application is used in determining similarities that exist between data points and requires less or no information to analyze the data. Despite K-means being a popular data clustering technique, it has some disadvantages tied to it. The first limitation is that the fixed number of clusters in this technique always requires being specified as an input to the algorithm [8]. Next, the determination of the clustering results leads to complications in achieving a good clustering outcomes [9]. Moreover, the data choice points collected randomly to determine the clusters could also lead to different final clusters, which implies that a rerun of the same may produce different results.

Nonetheless, K-means clustering algorithm is a good...
technique for obtaining critical data in a health facility since it enables the effectiveness of providing better health services based on patient records [10]. Many techniques have been developed to overcome each of the limitations of the K-means algorithm [9, 11–14].


In this paper, we propose an ensemble algorithm which uses Self Organizing Map (SOM) method with Genetic K-means clustering algorithm to automatically finds optimal number of segments in the data. Our algorithm also uses an optimization method Optimized Cluster Distance (OCD) which further improves the clustering results. This work is an extension of our previous research work on SOM clustering and visualization technique for clustering [18] and Genetic K-means with OCD for optimization [19]. SOM is a clustering and data visualization technique which is based on a neural network viewpoint [20]. The fundamental difference between SOM and K-means is that SOM produces number of segments based on predetermined topographic ordering relationship. During the training process, SOM uses each data point to update the segments and produces an ordered set of centroids for any given data set in the end. In other words, SOM overcomes the limitation of K-means algorithm by geometrically calculating the number of centroids in the data. Further Genetic K-means with OCD method overcomes the problem of general K-means by choosing best initial centroids using Genetic algorithm, as well as to maximize the distance between clusters by pairwise clustering using OCD. The rest of this paper is organized as follows. Section II presents a review of data mining techniques used in health care. Section III describes our proposed technique along with other clustering techniques used in this paper for comparison. Section IV provides a description of the dataset along with the experimental setup and Section V provides a discussion of the results as well as visualization of insights from clusters. Finally, Section VI concludes this paper along with future extension of our research.

2. Literature Review

Clustering techniques, though a fairly frequent field of study, is a very important field that benefits professionals such as computer scientists, pattern recognition experts, and statisticians [7]. Hence, a lot of research work has been conducted in relation to this technique. In [6], the benefits of K-means clustering algorithms are summarized in the introductory part of the work. It is indicated that this technique is one of the best that a data analyst may apply in the investigation of new data sets since it is algorithmically simple, provides efficient answers in a wide variety of data sets, and is relatively robust in nature.

Similarly, [7] states that K-means can be programmed with ease and is computationally economical. As such, allows processing of large samples of data, such as those of health records, on a computer. On the other hand, Haraty et al. [4] highlight its shortcoming through observation that an algorithm with improved initial centroids can avoid the random allocation of centroids through the use of the G-means variation of K-means. G-means uses a greedy algorithm to choose the initial centroids and then adjusts the centroids over multiple passes.

Kanungo et al. [21] compares the operations of K-means clustering technique and the Lloyds clustering technique. He indicates that K-means clustering is distinct from the Lloyd clustering because K-means tend to be more efficient with its use of information at every step it is applied.

Credit cards represent an enormous source of data and an important application for data mining. S. Fashoto et al. [22] discusses ways to use K-means clustering to improve credit card security and detect fraud.

As [8] states, determination of the number of groups of patients in a health facility is a significant problem that requires high precision. A. Likas et al. [9] indicates that one way to determine the appropriate figure of cluster k is by running the algorithm with a variety of k then choosing a k with the best result clusters. However, in as much as a rerun of the same may not produce the same results, the K-means technique is the most viable clustering method to apply in such cases [12, 23].

Another way to cluster data without a predetermined number of clusters is affinity propagation. Frey and Dueck [11] discuss how affinity propagation can be superior to the vertex substitution heuristic for large data sets.

K-means clustering can also suffer from the weakness of being restricted to numeric values. Huang and Zhexue [14] describe algorithms for doing K-means clustering using category data in addition to numeric data. Experiments show that these algorithms are efficient when the data set contains half a million objects.

Clustering can become a more difficult task when the data is in high dimensions. Vijendra [23] offers a review of clustering algorithms that look for clusters in subspaces in situations where clusters might not be found in the full data.
Data clustering can also be represented as a graph cut problem where each data point is a vertex and the edges are defined by the affinity metric between the points. A new algorithm for this kind of clustering is presented in [24].

One major alternative to K-means is support vector machines, but they quickly become slow with large data sets. H. Yu et al. [5] details a technique to make this approach more efficient when the data becomes large, and experiments show excellent scalability.

Fuzzy K-means allows clusters to overlap instead of being strictly defined. Sivarathri and Govardhan [25] explains how using a fuzzy approach can give superior results to a ordinary K-means technique.

Instead of using a single point to represent each cluster at the centroid of the cluster, it is possible to represent clusters using multiple medoids. Y. HE et al. [13] details a clustering algorithm that uses several weighted medoids within each cluster to represent the cluster to create a more precise representation of the underlying structure.

Bandyopadhyay [12] reviewed the possibility of using a genetic algorithm to assist with clustering. This approach has been a popular field of research and it can be applied with either a fixed number of clusters or a variable number of clusters.

M. de Hoon et al. [26] describes an open-source software library for various clustering techniques, including K-means, hierarchical clustering, and self-organizing maps. They based it on Michael Eisen’s Cluster program.

I. Yoo et al. [3] explain the history of data mining in the healthcare industry, including an overview of data mining and statistics and guidelines for using data mining in classification and association problems. We see the many purposes that data mining has been put to in medicine and the problems that limit the clinical use of data mining.

3. Methodology

This section describes the traditional K-means clustering algorithm, Genetic algorithm, Self-Organizing Map (SOM), and our proposed Optimized Cluster Distance (OCD) based SOM-Genetic K-means algorithm in detail.

3.1. K-Means Clustering

K-means is an unsupervised learning algorithm for partitioning data into clusters [27] [28]. Let \((x_1, x_2, \ldots, x_n)\) be a set of d-dimensional vectors and \(k (\leq n)\), then K-means tries to partition the \(n\) points into \(k\) clusters such as \(S = S_1, S_2, \ldots, S_k\) with minimized within-cluster sum of squares (WSS). The within-cluster sum of squares is the sum of the squares of the distances between each point in a cluster and the center of the cluster. For each \(i\), let \(c_i\) be the centroid of \(S_i\), then K-means has the following objective function:

\[
\arg \min_S \sum_{i=1}^{k} \sum_{x \in S_i} ||x_i - c_i||^2
\]  

3.2. Genetic Algorithm (GA)

Genetic algorithms are part of the larger class of algorithms called evolutionary algorithms which perform optimizations by means of varying genetic features. These features are varied by mutation, inheritance, crossover of attributes, and then selection of the most optimized individuals [29].

A Genetic algorithm is a randomized algorithm that allows each generational group to develop without direction, and then assesses the groups for fitness according to the objective. The attributes of each individual are mutated and spread into the next generation based on how well the individual satisfies the objective, until algorithm produces successive populations with greater and greater fitness levels up to a maximum number of generations.

3.3. Self Organizing Map (SOM)

The Self Organizing Map (also known as Kohonen map) is a valuable tool for finding clusters in complex data using unstructured artificial neural networks [20]. The SOM handles topological planning and preservation using a non-linear mapping design in a high dimensional data space. Each map uses a regular grid to define a neighborhood for each node. The neighborhood can then be used to propagate changes in a node to nearby nodes.

Let the input vector be \(x = (x_1, x_2, \ldots, x_p)\) and let the weight vector for node \(l\) be \(w_l = (w_{l1}, w_{l2}, \ldots, w_{lp})\) where \(w_{lj}\) indicates the weight assigned to input \(x_j\). Each datum is randomly chosen and the closest node is called the Best Matching Unit (BMU), then the BMU is moved closer to the datum. The distance between a node and a datum is measured by Euclidean distance, though other metrics could be used. The amount that each node moves is determined by the learning rate.

Let \(d(x, w_l)\) be the distance between input vector \(x\) and the weight vector for node \(l\).

\[
d(x, w_l) = \left( \sum_{j=1}^{p} (x_j - w_{lj})^2 \right)^{\frac{1}{2}}
\]  

Let \(w_l(s)\) be the weight vector for the \(s^{th}\) node on the \(l^{th}\) step of the algorithm. Let \(x(i)\) be the input vector for the \(i^{th}\)
training case. On each step, a training datum \( i \) is selected, and the index \( q \) of the BMU is determined by

\[
q = \arg \min_i \| w_q(s) - x(i) \| \quad (3)
\]

Let \( \alpha^s \) be the learning rate for the \( s^{th} \) step, then the new weight vector for the BMU on the next step is determined by

\[
w_q(s+1) = w_q(s)(1-\alpha^s) + x(i) = w_q(s) + \alpha^s(x(i)-w_q(s)) \quad (4)
\]

Once training is complete, data can be partitioned by the nodes according to the BMU of each datum.

### 3.4. Proposed Algorithm

We propose an efficient ensemble K-means clustering algorithm which combines SOM algorithm to find optimal number of segments and further enhanced it using Genetic algorithm and Optimized Cluster Distance (OCD) method which involves re-clustering of data with new centers and try to increase the between cluster distances (BSS) and decrease within cluster distances (WSS). Figure 2 shows the flow of our algorithm.

The algorithm starts by taking an input data set and processed it using SOM model. The key feature to SOM is that it transforms the original input data into topological features and preserves on the map. Inside the SOM model, the algorithm initialise all node weight vectors randomly and find the “Best Matching Unit” (BMU) in the map to get the most similar node together. Similarity is calculated using the euclidean distance formula for all the nodes within the neighborhood of the BMU. Since SOM is an iterative algorithm, it adjusts weights of nodes in the BMU with respect to learning rate until the solution converges and weight vectors becomes constant.

Once the conversion is made, the number of BMU indicated the optimal number of segments in the data. At this point, we use some visualization to get useful insights from each cluster and report the analysis. Once the data is transformed and optimal number of segments are calculated, our proposed algorithm goes to the next stage and transformed data is passed into Genetic K-means algorithm with OCD method to calculate the final clustering results. Genetic algorithm starts by generating initial population based on \( k \) provided as a parameter for number of clusters. Then it compute the fitness of initial population Mean Square Error (MSE). The MSE calculates the distance between the cluster centers and remaining data points.

\[
MSE = \frac{1}{n} \sum_{i=1}^{n} (x_i - c)^2 \quad (5)
\]

where \( c \) is the center from generated population.

\[
Fitness = \frac{1}{MSE} \quad (6)
\]

The algorithm then goes to the next generation or the next iteration if we want to get the better fitness than the fitness from previous generation. In that case, genetic algorithm performs selection, crossover, and mutation to generate new population. Process will stop if the best fitness value seems not to be changed in next generation.

Once Genetic algorithm outputs the best centroids, we run K-means algorithm on those centroids and get the clustering results. At this point, we introduce our novel technique called Optimized Cluster Distance (OCD) which further process the K-means clusters and try to minimize the within cluster variance and maximize distances between clusters. This is done by first calculating a pairwise distance between all clusters. Then the algorithm picks all pairs of clusters which have distances smaller than overall average of cluster distances. Once the pair of clusters are selected, the algorithm re-clusters them for specified number of iterations using the same K-means and GA algorithms. Since at this point we are clustering a small subset of data, the algorithm easily finds the partitions which are more distant to each other as compared to the partitions before OCD process. The pseudocode of our algorithm is stated in Algorithm 1.

### 4. Experimental Evaluation

#### 4.1. Dataset Description

This section contains a description of the datasets and experimental setup of this paper. The UIC Machine Learning Repository supplied two datasets on liver disease and heart disease [30]. The liver dataset comes from BUPA Medical Research Ltd and contains seven blood test variables which are considered to be sensitive to liver disease from excessive alcohol consumption. The data includes 345 patients with labels indicating whether the patient has a liver disease. Details can be found in Table I.

Heart disease dataset comes from Cleveland Clinic Foundation and contains fourteen heart disease variables. The data includes 303 patients with labels indicating if the patient has heart disease. Details of the heart disease dataset can be found in Table II.

#### 4.2. Performance Metrics

Both datasets contain the final diagnosis for each patient. The performance of our clustering algorithm is evaluated by metrics related to classification accuracy and visual insights to show the correctness of the results. Using the known class of each patient, each cluster is assigned to the class that is the most frequent in the cluster and we record the fraction of individuals that actually have that class in that cluster. We average the accuracy across all clusters, weighted by the number of records in the cluster. In addition, some
Fig. 1: Optimized K-Means-SOM-GA-OCD Algorithm

Input: Input dataset D with n features, size of grid W with i and j as dimensions, learning rate α

Output: Output dataset with k cluster labels

procedure SOM –GENETIC K–MEANS –

1: while α ≥ 0 do
2: for each x ∈ D do
3: for each wij ∈ W do
4: Calculate \( d_{ij} = \|x – w_{ij}\| \)
5: Select BMU that minimizes \( d_{ij} \)
6: Update each weight vector \( w_{ij} \) ∈ W
7: Decrease α
8: end for
9: end for
10: end while
11: intermediate Outputs: (i) SOM Topological Data
12: TData (ii) Optimal number of clusters k

procedure K–MEANS–GA–OCD

13: i = 1
14: while i ≤ N do
15: fitness = GA(population(i), TData)
16: if fitness(i) > fitness(i + 1) then
17: centers = GA-Centers(TData, k)
18: end if
19: end while
20: clusters = K-Means(TData, centers)
21: dist = PairwiseDistance(centers)
22: for each \( d_{xy} \) ∈ dist ≤ average(dist) do
23: while max iterations do
24: centers = GA-Centers(TData, 2)
25: newclusters = K-Means(TData, centers)
26: if wssnew ≤ wssxy then
27: \( cluster_{xy} = cluster_{newy} \)
28: \( cluster_{xy} = cluster_{newy} \)
29: end if
30: end while
31: end for
32: end procedure

TABLE I: Dataset Description of Liver Dataset

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mcv</td>
<td>mean corpuscular volume</td>
</tr>
<tr>
<td>alkphos</td>
<td>alkaline phosphatase</td>
</tr>
<tr>
<td>sgpt</td>
<td>alanine aminotransferase</td>
</tr>
<tr>
<td>sgot</td>
<td>aspartate aminotransferase</td>
</tr>
<tr>
<td>gammagt</td>
<td>gamma-glutamyl transpeptidase</td>
</tr>
<tr>
<td>drinks</td>
<td>alcoholic beverages drunk per day</td>
</tr>
<tr>
<td>selector</td>
<td>class label for liver disease</td>
</tr>
</tbody>
</table>

Attribute shows the feature names and description explains each feature of liver disease dataset.

TABLE II: Dataset Description of Heart Dataset

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>age in years</td>
</tr>
<tr>
<td>sex</td>
<td>patient gender</td>
</tr>
<tr>
<td>cp</td>
<td>chest pain type</td>
</tr>
<tr>
<td>trestbps</td>
<td>resting blood pressure</td>
</tr>
<tr>
<td>chol</td>
<td>serum cholesterol</td>
</tr>
<tr>
<td>fbs</td>
<td>fasting blood sugar</td>
</tr>
<tr>
<td>restecg</td>
<td>resting electrocardiographic results</td>
</tr>
<tr>
<td>thalach</td>
<td>maximum heart rate</td>
</tr>
<tr>
<td>exang</td>
<td>exercise induced angina</td>
</tr>
<tr>
<td>oldpeak</td>
<td>ST depression</td>
</tr>
<tr>
<td>slope</td>
<td>slope of the peak exercise ST segment</td>
</tr>
<tr>
<td>ca</td>
<td>number of major vessels</td>
</tr>
<tr>
<td>thal</td>
<td>exercise test</td>
</tr>
<tr>
<td>num</td>
<td>diagnosis of heart disease</td>
</tr>
</tbody>
</table>

Attribute shows the feature names and description explains each feature of heart disease dataset.

Visualizations were created to show the results in each cluster and help professional to make use of it.

Two additional metrics, Precision and Recall, were further used to evaluate our method and allow us to compare it with other methods. The Precision metric is the ratio of true positives to the number of all positives, the fraction of patients we diagnosed as having a disease that actually had the disease. Let \( TP \) be the number of true positives and let \( FP \) be the number of false positives, then

\[
Precision = \frac{TP}{TP + FP}
\]

The Recall metric is the ratio of true positives to the number of all people with the disease. Let \( FN \) be the number of false negatives, the people diagnosed as having the disease who did not have the disease, then

\[
Recall = \frac{TP}{TP + FN}
\]

5. Results and Discussions

5.1. Cluster Representation

We perform tests at each step of the algorithm. Figure 3 shows a heatmap from the SOM generated in the first stage of liver disease data using a 6 x 6 grid of nodes. This number of nodes is selected based on the statistics provided by other researchers who estimate that each node can be assigned 10 data points [31]. Dividing the 345 data points by 10 gives us roughly 35 and the closest square number is 36. The SOM algorithm finds the optimal number of clusters by taking each variable separately. We identified 5 clusters by exploring the heatmaps for all variables in Figure 3, with each cluster being indicated by a different color. The heatmap also shows the relationship between variables. For example, we can see that \( alkphos \) and \( sgpt \) variable heatmaps have an inverse relationship in many places of
The figure shows the process of ensemble SOM Genetic K-means OCD clustering algorithm (a) SOM algorithm to get optimal number of segments (b) Genetic algorithm to get optimal centroids (c) K-means algorithm with OCD optimization to get clusters (d) Visualization stage to get the valuable insights from data.

Heatmap, so we infer that patients with a low value of \( \text{sgpt} \) usually have a high value of \( \text{alkphos} \) and vice versa. The heatmaps are side by side, so they can be used to illustrate the different variables and their characteristics.

We run the genetic K-means algorithm on the SOM Topological data with the optimal number of clusters \( k \) found in first stage. Figure 4 has the final clustering output for five clusters. There are five different clusters in five different colors and separated by black lines. The same process for the heart disease dataset found four clusters identified by the SOM algorithm.

5.2. Cluster Evaluation

As explained in Section IV-B, we can find a weighted average accuracy for our clustering output using class labels. In the liver disease dataset, the variable \( \text{selector} \) is the class which indicates if a patient is infected. To evaluate a cluster, we took the cluster to represent the most frequent class present in it, then the accuracy of the assignment is evaluated by the percentage of records within the cluster that match the cluster’s class. The accuracy is averaged across all clusters and weighted by the number of records in each cluster. Let \( n_i \) be the number of records in cluster \( i \). Let \( \text{Accuracy} \) be the classification accuracy for cluster \( i \). Let \( N \) be the total number of records in the dataset, then the weighted average accuracy is as follows

\[
\sum_{i=1}^{k} \frac{\left(n_i \times \text{Accuracy}\right)}{N}
\]

Table III shows the accuracy for the proposed algorithm and some other methods on the two datasets. Our testing indicates that our proposed algorithm improves upon other methods in classification accuracy and is able to generate useful insights. Figure 7 reports Precision and Recall for our proposed method using liver disease dataset and compares with other methods. Our test experiment indicates that our proposed algorithm improves upon other methods in these metrics.

<table>
<thead>
<tr>
<th>Method</th>
<th>Liver Disease Data</th>
<th>Heart Disease Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOM-GA-KM-OCD</td>
<td>75.25</td>
<td>72.88</td>
</tr>
<tr>
<td>SOM-GA-KM</td>
<td>73.84</td>
<td>69.90</td>
</tr>
<tr>
<td>KM</td>
<td>69.15</td>
<td>66.27</td>
</tr>
<tr>
<td>DBSCAN</td>
<td>67.66</td>
<td>61.45</td>
</tr>
</tbody>
</table>

The number shows the weighted classification accuracy of different algorithms in percentage using liver and heart disease datasets.
Fig. 3: Heatmaps for visualisation of possible Self-Organizing Maps
The figure shows the visualization of SOM for each variable in Liver disease dataset. Each plot is scaled to their real values in order to see the correct relationships between variables. The color bar on y-axis shows the min and max value for each feature.

Fig. 4: Cluster representation of liver disease data using SOM Genetic K-means with OCD algorithm.
The figure shows the visualization of different clusters generated by SOM Genetic K-means along with OCD algorithm using liver disease data when number of clusters are set to 5. Each color represents a different cluster/segment.

5.3. Visual Insights Application
We processed the clustering results to gain more visual insight. Figures 5 and 6 show a comparative density plot for clusters 2 and 3 in the liver disease dataset. This plot illustrates how each cluster preserves its uniqueness as compared to other clusters. Figure 5 demonstrates that patients in cluster 2 have a high mean value of alkphos with respect to the full dataset. The uniqueness of cluster 2 is also apparent in each of its other variables. Thus, we can see how these visualizations help to explain the variables in the dataset. We also compared Figures 5 and 6 and saw that cluster 3 patients showed a lower mean on gammagt and cluster 2 patients showed a higher mean relative to the full dataset. We can infer that cluster 3 contains patients who mostly have low gammagt value relative to the overall population.

6. Conclusion
In this paper, we propose an efficient ensemble method to investigate health care data. Our method uses Self Organizing Map (SOM) to infer optimal number of clusters at the initial stage, then apply Genetic K-means algorithm with Optimized Cluster Distance (OCD) method to cluster health care data for knowledge discovery. With the empirical evaluation, we show that our proposed algorithm outperforms other existing methods. We also present a visual application using density distributions of the training variables within each cluster to build a meaningful picture of the cluster characteristics. For future work, we would like to use our method in different domains to solve different challenges such as document clustering, image clustering, social network analysis, network security etc. and compare it with the state-of-the-art methods.

Acknowledgment
The authors would like to thank the Department of Computer Science at Bowie State University.

References
Fig. 5: Density plot using liver disease data to get insights on cluster 2
The figure shows the comparative density plot of a cluster 2 for each feature separately against other clusters i.e whole population using liver disease data.

Fig. 6: Density plot using liver disease data to get insights on cluster 3
The figure shows the comparative density plot of a cluster 3 for each feature separately against other clusters i.e whole population using liver disease data.

[5] H. Yu, J. Yang, J. Han, and X. Li, “Making svms scalable to large data sets using hierarchical cluster indexing,” *Data Mining and Knowledge Discovery*, }
KMeans-GA-OCD), K-Means, DBSCAN and SOM-GA-KMeans algorithm on liver disease dataset.


**Fig. 7: Precision Recall Comparison Of Proposed Algorithm**

**References**


Ahmed Alsayat is pursuing his Ph.D. in the Department of Computer Science at Bowie State University. In 2006, Alsayat received a B.E in Computer from Aljouf University, Saudi Arabia. Alsayat graduated from Florida Institute of Technology with a M.S in Computer Science in 2009. From the same school, he received his second M.S in Engineering Management in 2009. In 2010, Ahmed Alsayat worked as a lecturer in the department of computer science at Aljouf University which is sponsoring his post graduate studies. He was also working as the manager of the network and operating systems department in the IT of Aljouf University. His research interests are including data mining, machine learning and Web mining.

Hoda El-Sayed is a Professor at Bowie State University where she serves as Director of Doctoral Program in the Department of Computer Science. She earned her D.Sc. in Computer Science at The George Washington University in Washington, DC. Her specialties are Parallel and Distributed Algorithms, High-Performance Computing and Applications, Parallel Processing Systems. Dr. El-Sayed has published several refereed research publications in this area. Her work has been supported by federal agencies as the DoD and NSA. Dr. El-Sayed has served in many editorial conferences. Dr. El-Sayed has been awarded The University System of Maryland Board of Regents Faculty Award for Excellence in Innovation in March 2016 and also awarded Bowie State University College of Arts and Science Outstanding Faculty Award in April 2016. Dr. El-Sayed is a Senior Member of IEEE and member of the Association for Computing Machinery.
Meta Path-Based Temporal Link Prediction in Heterogeneous Evolving Networks

Alper Ozcan¹, Sule Gunduz Oguducu¹
Istanbul Technical University, Turkey

Abstract

Link prediction is a significant task in social network analysis. Most existing link prediction studies focus only on homogeneous networks, where all nodes and links are related to the same type. However, many real world social networks include different link and node types. Moreover, heterogeneous social networks are very dynamic, which may lead the multi-typed nodes and edges to appear or disappear in the future. Therefore, recently added or removed links in future social networks is very important for understanding the evolution of social networks. In this article, we improve the meta path-based similarity measure-PathSim by combining temporal information. We propose a link prediction model for temporal heterogeneous networks. The proposed model first calculates the heterogeneous similarity measure score (PathSim score) for different time periods. Then, effective time series forecasting model, ARIMA, is applied for the prediction of the future node similarity scores by using past node similarities. The proposed model is tested on two real world temporal heterogeneous networks. It is shown that using time information with the meta path-based similarity measure-PathSim enhances the link prediction performance.

Keywords: Heterogeneous Social Network Analysis, Evolving Networks, Meta Path-Based Topological Features, Link Prediction, ARIMA

1. Introduction

Link Prediction has become one of the significant research topic in link mining and social network analysis tasks. Most of the existing link prediction methods [1], [2], [3], [4], [5] are created for homogeneous networks, where all nodes and links related to the same type. Furthermore, most of the existing link prediction methods [1], [2], [3], [4] ignore the dynamicity of the networks over time and predict the future links based on the static features of nodes and links. However, real-world networks have heterogeneous and dynamic structures which means that both multiple types of nodes and links in social networks could appear or disappear. These factors in total, makes link prediction a challenging task. For this reason, in this paper, we study the problem of link prediction in the temporal heterogeneous network. The proposed model tries to handle these challenges by analyzing how topological similarity measures evolve in the network over time.

Link prediction task in Social Network Analysis is best described as the task of predicting whether a link will form between two nodes in the future considering previous observed links in a social network [1]. A common approach for link prediction is to use static similarity topological measures to predict the future links. The static measures assume that they never vary over time which is not true in every case. Such measures assign scores to each pair of nodes which are then used to perform the link prediction task. Also, many studies in this field only perform the task by examining the snapshot of the network for a particular time period to predict the links, without considering the behavior of evolving links over time. The biggest disadvantage of these approaches is the ignorance of the time effect. In this context, we examine if temporal information may enhance the performance of the link prediction task. For example, consider the heterogeneous bibliographic network involve author node type and other types of nodes, such as papers and venues. In this heterogeneous bibliographic network, two nodes of type authors can be linked by co-author relationships and publication in same venue relationships. These different link types which influence each other have different semantic meanings and various link formation mechanisms. In order to find similarities between nodes, meta path-based similarity measure-PathSim [6] is used. PathSim is defined as the meta path-based similarity measure, where a meta path contains a sequence of relations specified between different node types [7]. According to the PathSim measure, two nodes are similar if there are many paths between them. Furthermore, this heterogeneous network and the similarity between the two authors can vary over time. Intuitively, the more recent links make more contribution to the similarity than the older ones. In order to understand the impacts of links formed at different time periods, time series model ARIMA [8] is created for each pair of nodes.

In this article, we aim to understand temporal behavior of heterogeneous links and how this time-varying information can be used to predict future heterogeneous links. To this end, we propose a novel link prediction model which employs temporal network information in order to predict heterogeneous links in evolving heterogeneous networks.
First, evolving heterogeneous network is divided into time-sliced snapshots for each time period, where each snapshot represents the states of the network. Then, proposed model computes the meta path-based topological similarity measure (PathSim) score values for all node pairs in each snapshot. After that, time series model is built for each pair of nodes to estimate the future node similarity scores between node pairs. Lastly, these estimated node similarity scores are used as input to supervised link prediction models and link prediction is performed based on these estimated node similarity scores. The link prediction performances of proposed model and baseline models are evaluated under different circumstances by means of different AUC measures. The experiments showed that our proposed model outperforms the baseline models. The contributions of this article are as follows:

- Proposed a novel model to extend the typical usage of meta path-based similarity measure PathSim by integrating temporal information.
- Effective usage of time-varying characteristics of heterogeneous networks for link prediction.
- Experiments on two temporal real-world heterogeneous datasets show that time series forecasting on the meta path-based similarity scores between nodes improved the link prediction performance.

The rest of the article is organized as follows. Related works are given in Section 2. We present the background information on heterogeneous similarity measures used in this paper. Detailed description of the proposed temporal heterogeneous methods is given in Section 4. The experimental results are presented in Section 5. Finally, the conclusion can be found in Section 6.

2. Related Work

There are different methods which have been implemented in the domain of link prediction problem. These methods can be classified into two major types based on the network type used: (1) Link prediction methods for homogeneous networks and (2) Link prediction methods for heterogeneous networks.

One of the earliest link prediction research shows that the link prediction is mainly based on unsupervised methods [1], in which various similarity measures are created from topological similarity measures structure of the homogeneous networks. The main assumption of the topological similarity measures is that, node pairs are more likely to have a link in the future if they have many common features. In the topological methods, the pairs of non-connected nodes are ranked according to the selected measures and the top ranked non-connected nodes are predicted to be linked in the future. Topological similarity measures are divided into three types based on the information used by the similarity measures: (1) local measures, (2) global measures and (3) quasi-local measures. Local similarity measures are restricted to the nearest neighbors of nodes and do not cover the entire network. Local similarity measures include Common Neighbors [1], Salton Index [2], Jaccard Index [9], Hub Depressed Index [2], Adamic/Adar [10], Leicht-Holme Newman Index (LHN1) [11] and Preferential attachment [12]. Whereas the global measures, unlike the local measures, require the entire topological information. Although the global measures can provide high link prediction performance in terms of accuracy than the local measures, global ones are negatively affected by the computational complexity. Global measures have high time complexity and these measures are commonly impractical for large-scale networks. Global similarity measures include Katz Index [13], PropFlow [14] and Random Walk with Restart (RWR) [2]. Generally, quasi-local measures do not require global topological information, however they extract more information than local measures. Quasi-local measures include Local Path Index [2], Local Random Walk [15], Superposed Random Walk [2]. On the other hand, supervised methods are also proposed to integrate different characteristic of network via training data sets [16].

Most of the previous study on link prediction only depends on the static snapshot of the network. However, social networks are very dynamic, since new links and nodes appear in the network over time. Understanding the dynamics that employ the evolution of social network is a complex problem, specially due to a large number of changing characteristic features of the network. Since the effect of temporal behavior of the network structure are neglected by static similarity measures, different temporal methods have been proposed to predict the future links. Tylenda et. al. (2009) proposed the local probabilistic model to implement a method for time-aware link prediction [5]. Oyama et. al. (2011) introduced cross temporal link prediction method to predict the links in different time frames [17]. Moradabadi and Meybodi presented a link prediction method based on temporal similarity metrics and Continuous Action set Learning Automata (CALA). Time Series based models such as AR, MA, ARIMA are also proposed for evolving networks to predict link occurrence probabilities at a particular time taking into consideration temporal evolutions of link occurrences and similarity measures [18]. Lastly, Ozcan and Gunduz-Oguducu (2015) performed Multivariate Time Series model called Vector Autoregression (VAR) to combine the correlations among similarity measures and link occurrences information simultaneously [19].

All the previous studies in link prediction focus on homogeneous networks. However, real-world networks
have heterogeneous relations and entities, which makes the link prediction a very difficult task. Similarity measures defined on heterogeneous networks have been proposed recently. Sun et al. (2011) specified the meta paths and assign any weight to them. They also introduced meta path-based similarity measure called PathSim in order to compute pair-wise similarities between nodes [7]. Davis et al. (2011) described a measure called multi-relational link prediction (MRLP) for heterogeneous information networks that is a weighted extension of the Adamic/Adar measure [20]. Yong et al. (2014) proposed a link prediction method for heterogeneous network based on tensor factorization [21].

In this article, we propose an effective time series forecasting model which is applied for the prediction of the future node similarity scores by using past node similarities. We extend the utilization of heterogeneous similarity measures for link prediction to the evolving heterogeneous networks. Variation of similarity measure values over time are analyzed using time series and in this way future similarity values are estimated. First, our proposed model computes pair-wise similarities between nodes by various graph-based heterogeneous similarity measures and estimates the future node similarity scores between node pairs. Then, these estimated node similarity scores are employed as input to supervised link prediction models and link prediction is implemented, based on these estimated node similarity scores. By using the time series model, proposed model investigates how the relationships between nodes change over time and how the relationship between two nodes is affected by other nodes. As a result, our approach has the advantage of both the time series forecasting model and the heterogeneous node similarity measures.

3. Background

In this section, we have provided the definitions of meta path used in this paper and background about PathSim similarity measure and other measures.

3.1 PathSim

PathSim is described as the meta path-based similarity measure, where a meta path includes a sequence of relations between different node types [7]. Heterogeneous bibliographic network created in this study contains multiple types of nodes, such as authors (A), venues (V) and papers (P), as well as different types of links specifying different relationships among these nodes, such as "write/written by", "cite/cited by" and "publish/published by" relations.

### Table 1. Meta Paths Between Authors in the HEP-Th Network

<table>
<thead>
<tr>
<th>Meta Path Definition</th>
<th>Semantic Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>A – P – A</td>
<td>$a_i$ and $a_j$ are co-authors</td>
</tr>
<tr>
<td>A – P → P – A</td>
<td>$a_i$ cites $a_j$</td>
</tr>
<tr>
<td>A – P ← P – A</td>
<td>$a_i$ is cited by $a_j$</td>
</tr>
<tr>
<td>A – P → V → P – A</td>
<td>$a_i$ and $a_j$ publish in the same venues</td>
</tr>
<tr>
<td>A – P → P → P – A</td>
<td>$a_i$ cites papers that cite $a_j$</td>
</tr>
<tr>
<td>A – P ← P ← P – A</td>
<td>$a_i$ is cited by papers that are cited by $a_j$</td>
</tr>
<tr>
<td>A – P → P → P – A</td>
<td>$a_i$ and $a_j$ cite the same papers</td>
</tr>
<tr>
<td>A – P ← P ← P – A</td>
<td>$a_i$ and $a_j$ are cited by the same papers</td>
</tr>
</tbody>
</table>

Figure 1 shows the Bibliographic Network schema and example co-authorship meta path defined over the network schema : APA. In this network schema, "write" and "written by" relations are formed between authors and papers, "publish" and "published by" relations are formed between authors and venues, and so on. Example APA meta path denotes the co-authorship relation. Table 1 shows the meta paths between authors up to length 4 and the semantic meaning of each relation expressed by each meta path. In the bibliographic network case, we study the problem of predicting future co-authorship relation (target relation) between existing authors.

![Figure 1. Bibliographic Network Schema and Meta Path : APA](image)
Also, heterogeneous bookmarking network constructed in this study includes multiple types of nodes, such as users (U), resources (R) and tags (T), as well as different types of links indicating different relationships among these nodes, such as "bookmark/bookmarked by" and "contain/contained by" relations. Figure 2 shows the Bookmarking Network schema and example meta path defined over the network schema: URTRU. Example URTRU meta path denotes the same tags shared by two users. Also, Table 2 represents the meta paths between users up to length 4 and the semantic meaning of each relation denoted by each meta path. In the bookmarking network case, we study the problem of predicting future bookmarking of same resource relation (target relation) between existing users.

<table>
<thead>
<tr>
<th>Meta Path Definition</th>
<th>Semantic Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>U – R – U</td>
<td>$u_i$ and $u_j$ bookmark same resource</td>
</tr>
<tr>
<td>U – R – T – R – U</td>
<td>$u_i$ and $u_j$ use same tag</td>
</tr>
<tr>
<td>U – R – U – R – U</td>
<td>$u_i$ and $u_j$ bookmark same resource with the same users</td>
</tr>
</tbody>
</table>

Table 2, Meta Paths Between Users in the Delicious Network

\[ PathSim^t(u, v) = \frac{2 \times |\{p_u \rightarrow v : p_u \rightarrow v \in \mathcal{P}\}|}{|\{p_u \rightarrow u : p_u \rightarrow u \in \mathcal{P}\}| + |\{p_v \rightarrow v : p_v \rightarrow v \in \mathcal{P}\}|} \]  

where $p_u \rightarrow v$ is a path instance between node $u$ and node $v$, $p_u \rightarrow u$ is a path instance between $u$ and $u$. Multiple occurrences of a path instance are considered as the weight of the path instance, that is the product of weights of all the links in the path instance [7].

3.2 Path Count (PC)

Path count calculates the number of path instances between two nodes at a time period $t$ following a given path, denoted as $PC^t_R(u, v)$, where $R$ is the relation denoted by a meta path.[7]

3.3 Normalized Path Count (NPC)

Normalized path count is the calculation of the ratio of the number of paths between two nodes by their complete connectivity, and is specified as

\[ NPC^t_R(u, v) = \frac{PC^t_R(u, v) + PC^{t-1}_R(u, v)}{PC^t_R(u, *, *) + PC^{t-1}_R(*, v)} \]  

where $R^{-1}$ denotes the inverse relation of $R$, $PC^t_R(u, *)$ denotes the total number of paths following $R$ starting with $u$. [7]

4. Method

In this section, we introduce the structure and steps of our novel link prediction model for temporal heterogeneous networks. Firstly, we split into temporal heterogeneous network into time-sliced snapshots. Each snapshot $(G^t, t = 1, \ldots, T)$ identifies the state of the network at different time periods. The basic time series link prediction problem is determined as to predict the probability of link occurrence at a given point of time $t = T + 1$.

We propose ARIMA-Sim model that exploits univariate time series of different heterogeneous similarity measures. In proposed model, time series forecasting is employed for each pair of nodes in consecutive snapshots in order to estimate the future scores based on the past score values. Therefore proposed model extends the common usage of node similarities for link prediction. Also the baseline models include AR-Sim, MA-Sim, Sim-Last and Sim-All. AR-Sim and MA-Sim utilize univariate time series of similarity measures.
The MA model of order \( q \), written as MA(\( q \)), is defined as in Eq. 3, where \( \theta_1, ..., \theta_q \) are the parameters of the model and the \( \epsilon_{t}, ..., \epsilon_{t-q} \) are white noise error terms

\[
\hat{Y}_t = \epsilon_t + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q} \quad (3)
\]

The AR model of order \( p \), written as AR(\( p \)), is defined as in Eq. 4, where \( \phi_1, ..., \phi_p \) are the parameters of the model and \( \hat{Y} \) indicates the estimated output.

\[
\hat{Y}_t = \sum_{r=1}^{p} \phi_r Y_{t-r} + \epsilon_t \quad (4)
\]

The ARIMA model that includes AR and MA models, written as ARIMA(\( p,d,q \)), is defined as in Eq. 5.

\[
\hat{Y}_t = \sum_{r=1}^{p} \phi_r Y_{t-r} + \sum_{s=0}^{q} \theta_s \epsilon_{t-s} \quad (5)
\]

In time series models (ARIMA-Sim, AR-Sim, MA-Sim), similarity scores for each pair of nodes are used as an input in the time series models and thus, future score values are estimated. On the other hand, Sim-Last uses the links in the last time period and Sim-All examines all links for all time periods and considering that two nodes are connected if they have ever been connected in the past. In all these models, the abbreviation Sim means different similarity measure scores such as PathSim, Path Count, Normalized Path Count.

The proposed model is composed of three main steps:

- **First Step**: PathSim, Path Count and Normalized Path Count similarity measure scores are computed for different time periods.
- **Second Step**: The past similarity score values in different time periods are used as time series and forecasting of the future similarity values is achieved based on this model.
- **Third Step**: Supervised learning method is implemented to learn the weights associated with each similarity measure scores in estimating links.

As classifier, we use the implementation of Support Vector Machine (SVM) from the WEKA framework with default parameters. The features used in the SVM are the forecasted similarity measures between the pair of nodes which are estimated by the ARIMA-Sim and other time series models. Then, link prediction is performed by using the supervised learning method. Last, the performance of link prediction is evaluated in terms of AUC values.

5. Experimental Results

In this section, we explain the datasets employed in our experiments and indicate the results obtained by our proposed time series model for link prediction as compared to the baseline models.

5.1 Dataset

The HEP-Th\(^2\) Dataset is used in the experiments. This dataset has been widely used in link prediction studies. The HEP-Th dataset provides a wide ranging list of research papers in theoretical high-energy physics. The entire dataset contains 29,555 papers published between years 1987 and 2003 and 9,200 authors and 448 journals. These entities are connected by 352,807 citation, 87,794 co-authorship, 58,515 authorship, 20,816 publication links. We extract data from the year 1991 to 2003. Then, temporal network dataset is divided into time-sliced snapshots from the year 1991 to 2002 to create the time series model where each network snapshot is for one year and the links in the year 2003 are employed as the test set to be predicted. In this bibliographic network, the task is to predict the likelihood of co-authorship link type between authors in the future.

The other dataset used in the experiments is Delicious. The dataset is downloaded from Delicious between 2008 and 2010. This dataset consists of 1,358,522 bookmarking and 5,546,813 tagging relations between users. We extract resources (URL) and tags data from the month January, 2010 to December, 2010. Then, temporal network dataset is divided into time-sliced snapshots from the month January to November to create the time series model where each network snapshot is for one month and the links in the month December are used as the test set to be predicted. In this bookmarking network, the task is to predict the likelihood of bookmarking the same resource relation (target relation) between existing users in the future.

\(^1\)https://kdl.cs.umass.edu/display/public/HEP-Th
\(^2\)http://del.icio.us
In order to measure the link prediction performance, we apply Area Under Curve (AUC) measure since it is an important performance measure which has also been used in link prediction. Also, to perform the paired two sample t-test, which is commonly employed as a statistical hypothesis test analyzing the AUC measures [8], 10-fold cross-validation is used.

5.2 Experimental Results

The main purpose of the experiments is the evaluation and comparison of link prediction performance of our proposed model (ARIMA-Sim) with the baseline models (AR-Sim, MA-Sim, Sim-Last and Sim-All). We propose link prediction approach (ARIMA-Sim) based on the input employed for the time series analysis as explained in Section 4. The approach is implemented with three heterogeneous similarity measures (ARIMA-PathSim, ARIMA-PC, ARIMA-NPC). Here, the variation of node similarities are analyzed over time and thus future score values are estimated. Then, a supervised learning method is used to learn the best weights related to estimate the different topological measures scores in predicting links. In order to compare our proposed model with the baseline models, four groups of baseline models are included: AR-Sim, MA-Sim, Sim-Last and Sim-All. In baseline time series models (AR-Sim, MA-Sim) [8], similarity scores for each pair of nodes are used as input and future score values are hereby estimated. Also, Sim-Last employs the links in the last time period and it statically analyzes the network at the last time period only. Sim-All combines all snapshots of temporal network into a single network, then it considers all links for all time periods and it examines that two nodes are connected if they have ever been connected in the past.

Figure 3 includes the link prediction results on the HEP-Th dataset in terms of the AUC value. The results are grouped by the similarity scores. Figure 3 shows that ARIMA-Sim is the best forecasting model for HEP-Th dataset and the proposed model (ARIMA-Sim) outperforms the baseline time series models (AR-Sim and MA-Sim) by more than 5%. Also, the proposed model (ARIMA-Sim) outperforms by more than 15% compared to the baseline models Sim-Last and Sim-All. It can be asserted that the variation of heterogeneous similarity measures over time are more statistically significant of the link prediction than the heterogeneous similarities based on the last time period (Sim-Last), using heterogeneous similarities on the combined past data (Sim-All).

Figure 4 shows the link prediction results on the Delicious dataset in terms of the AUC value. Again, the proposed model (ARIMA-Sim) also outperforms by more than 10% compared to the baseline models Sim-Last and Sim-All, as it can be seen in Figure 4. Among the baseline models, using only the heterogeneous similarities of all times whose results are obtained from the Sim-All model performs better than the other static baseline model Sim-Last. This result shows that the link prediction performance of using heterogeneous similarities is increased by integrating all the past links. According to the comparison results of the AR-Sim and MA-Sim with the Sim-Last and Sim-All, AR-Sim and MA-Sim perform better than the other baseline models. Modeling the change of heterogeneous similarities over time can be the possible reason for this performance enhancement.
Also, we can say that modeling the change of heterogeneous node similarities over time, enhances the link prediction performance for these datasets. The proposed model ARIMA-Sim using PathSim measure score, provides the best results. Simple time series of heterogeneous node similarities (AR-Sim and MA-Sim) that analyze short-term change of similarity measures give the second best results.

Finally, according to the results we have achieved, when comparing the PathSim similarity measure with Path Count and Normalized Path Count measures, PathSim similarity measure that extracts more information about the underlying social structure, perform better than Path Count and Normalized Path Count measures. PathSim performs the best for all models because this measure considers the contribution of the meta paths between different node types and makes a much more accurate prediction than the other similarity measures.

5.3 Statistical Significance Tests

We employ 10-fold cross-validation for statistical significance test experiments. The results for HEP-Th and Delicious data sets are presented in Table 3 and Table 4, respectively. Standard deviation values which are between 0.01 and 0.06, are also provided in these tables (p < 0.05). As it can be seen, the statistical significance test results also show that PathSim measure performs the best for all forecasting models in both of the datasets. According to the comparison of the standard deviation values of each method, ARIMA-Sim with PathSim measure also provides the most stable prediction performance. This may be due to the fact that PathSim uses the meta paths between different node types and makes a much more accurate prediction than other similarity measures. Sim-Last model is the worst among the baseline models because it only considers the last observed links in the last time period. According to the time series models’ results, ARIMA-Sim model can investigate long-term evolution of similarity measures which is crucial in link prediction for temporal networks and it provides better results against the other simple time series models AR-Sim and MA-Sim.
Table 3. 10-fold cross-validation comparison results on HEP-Th dataset

<table>
<thead>
<tr>
<th>PathSim</th>
<th>PC</th>
<th>NPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARIMA-Sim</td>
<td>0.83±0.01</td>
<td>0.75±0.02</td>
</tr>
<tr>
<td>AR-Sim</td>
<td>0.77±0.03</td>
<td>0.71±0.04</td>
</tr>
<tr>
<td>MA-Sim</td>
<td>0.74±0.02</td>
<td>0.72±0.05</td>
</tr>
<tr>
<td>Sim-Last</td>
<td>0.65±0.04</td>
<td>0.61±0.04</td>
</tr>
<tr>
<td>Sim-All</td>
<td>0.68±0.03</td>
<td>0.63±0.04</td>
</tr>
</tbody>
</table>

Table 4. 10-fold cross-validation comparison results on Delicious dataset

<table>
<thead>
<tr>
<th>PathSim</th>
<th>PC</th>
<th>NPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARIMA-Sim</td>
<td>0.79±0.01</td>
<td>0.72±0.03</td>
</tr>
<tr>
<td>AR-Sim</td>
<td>0.75±0.04</td>
<td>0.71±0.03</td>
</tr>
<tr>
<td>MA-Sim</td>
<td>0.68±0.03</td>
<td>0.65±0.04</td>
</tr>
<tr>
<td>Sim-Last</td>
<td>0.60±0.04</td>
<td>0.57±0.06</td>
</tr>
<tr>
<td>Sim-All</td>
<td>0.62±0.04</td>
<td>0.57±0.05</td>
</tr>
</tbody>
</table>

6. Conclusions

Link prediction problem in heterogeneous network is different from the link prediction problem in homogeneous network, which generally aims at predicting the future links of the same type in the next time period. Because real world networks have a highly dynamic and heterogeneous structure, which make link prediction a more challenging task, we study the problem of link prediction in the temporal heterogeneous network in this paper. In this paper, we proposed a novel temporal heterogeneous link prediction model based on ARIMA time series forecasting model. We enhance the meta path based similarity measure PathSim by combining temporal information. The proposed model first calculates the heterogeneous similarity measure score (PathSim score) for different time periods. Then, effective time series forecasting model, ARIMA, is applied for the prediction of the future node similarity scores by using past node similarities. Lastly, link prediction is performed by using the supervised learning method. Our experimental results show that modeling of the change of past node similarities and the effect of change over time are very important to understand the underlying mechanisms of temporality. In terms of AUC performance, the proposed method results in an increase of 8-20% compared to the baseline models. In the future research, we will aim to extend our proposed model to the different social networks.

References


Alper Özcan is currently pursuing his Ph.D. in Department of Computer Engineering at Istanbul Technical University. Alper Özcan has received a B.S. in Computer Engineering from Istanbul Technical University, Istanbul Turkey in 2007, and an M.S. in Computer Engineering from Istanbul Technical University, Istanbul, in 2010. His research interests include social networks, data mining and machine learning.

Dr. Sule Gunduz Oguducu is an associate professor of Computer Engineering in the Department of Computer Engineering at the Istanbul Technical University. She is teaching undergraduate and graduate level courses. She is the head of the Learning from Data Research Laboratory at the Department.

Dr. Sule Gunduz Oguducu has received a B.S. in Electronics and Communication Engineering from Istanbul Technical University, Istanbul Turkey in 1991, and an M.S. in Biomedical Engineering from Bogazici University,
Istanbul, in 1995. She carried out her Ph.D. studies from 1999 to 2003 at the Istanbul Technical University. During her doctoral study, she was a visiting research scholar at the University of Waterloo, Waterloo, Canada, in the Database Research Group. In 2003, she won Siemens Excellence award for her PhD thesis.

Dr. Gündüz Oguducu’s research interests are data mining, Web mining and recommender models. In these fields, she is a reviewer and co-author of various publications in prestigious journals and conferences. In 2010, she published a book titled "Web Page Recommendation Models: Theory and Algorithms" (Morgan and Claypool Publishers). She is a co-editor of a book titled “Social Networks: Analysis and Case Studies” published in 2014 by Springer in their Lecture Notes in Social Networks series. She has served as the Program Chair of the Data Mining and Business Intelligence Workshop held in conjunction with ICDE 2007 and ISCIS 2008. In 2010, she founded the IEEE Turkey WIE Affinity Group.
Prediction Interval of Cumulative Number of Software Faults Using Back Propagation Algorithm

Begum Momotaz* and Tadashi Dohi
Hiroshima University, Japan

Abstract

In this paper, we apply the well-known back propagation algorithm for feed forward neural network architectures and the delta method to construct the prediction intervals of cumulative number of software faults, where the underlying fault count data are governed by the Poisson law. To deal with the Poisson count data in the neural network computing, we use three data-transform methods for pre-processing the Poisson data; Bartlett transform, Anscombe transform and Fisz transform. In numerical experiments with eight real software development project data sets, we evaluate the one-stage look ahead prediction interval in sequential software testing, and compare these data transform methods in terms of average relative error, coverage rate and prediction interval width.

Keywords: software reliability, fault prediction, prediction interval, back propagation algorithm, delta method.

1. Introduction

Our modern society depends on computer systems, so that there are many different applications, such as nuclear reactors, financial trading systems and medical systems, etc. Software controls our daily activities on computer systems. Failure may occur even when software is used. Nowadays, predicting software faults is an applicable issue and a major anxiety for software developers and engineers. In software development project, most of software engineers set goals to provide high quality software. Especially reliability of software plays a vital role in software development process, because software reliability is one of the most fundamental but significant quality attributes [6][31][32][34][43]. Many quantitative methods such as reliability modeling, development cost prediction, release-decision analysis and fault-tolerance, have been developed in software engineering [14][15][20][21][42] and been based on software reliability growth models (SRGMs).

The major drawbacks of SRGMs are that their goodness-of-fit performance and the prediction performance strongly depend on the data, so it is always needed to select the best SRGM by estimating the latent number of faults in each software product. In general, SRGMs can be classified into different several classes, such as error seeding models, failure rate models, and nonhomogeneous Poisson process (NHPP) models [6][31][32][34][43].

On the other hand, there are two classes of SRGMs; analytical SRGMs and data-driven artificial neural network models. However, none of these models satisfy the requirement levels of software developers [38]. Furthermore prediction of software faults is an important measurement to find reliable software in the software operational phase. Sometimes professional managers use any predictor for software quality after release. Abaei et al. [2] explained software fault prediction based on different machine learning techniques such as decision trees, decision tables, random forest, neural network, naïve Bayes and distinctive artificial immune systems classifiers. They made a conclusion that random forest outperforms the other methods.

In recent years, artificial neural network approaches have proven to be a universal approximation for arbitrary nonlinear continuous function with arbitrary accuracy [7][8][10][11][18][19][26][27][28][29][30][36][40]. Besides, artificial neural network models need to specify the network architecture, but do not unrealistic assumptions with respect to software failure data and tend to be superior to the classical SRGMs due to their freedom on model selection. They have been becoming an alternative to the traditional prediction method based on SRGMs [16]. Specifically, feed forward back propagation neural networks have shown their advantages over analytical models in prediction on the number of software faults.

Artificial neural network is a collection of several processing nodes called artificial neurons. These neurons are designed on the basis of study of the behavior of biological neuron. Basically it is good at modeling nonlinear relationships and interaction, while conventional statistical analysis in most cases implicitly assumes a linear relationship between independent variables and dependent variables. Neural network builds its own models through a learning process, to examine whether the relationships among variables are linear or not. Neural networks can also deal with missing or incomplete data, and can update the weights...
among input, output and intermediate nodes, so that even incomplete data can contribute to learning and can produce desired output results. Back-propagation is one of the most widely used neural network paradigms and has been applied successfully to many application studies in a broad range of areas [3][25][27][39].

Cai et al. [7] examined that handling datasets with ‘smooth’ trends is more effectiveness in the neural network approach than handling datasets with large fluctuations. They found that the neural network approach is much better in prediction than SRGMs. Sherer [35] predicted software faults in several NASA projects with neural networks [13][22][23]. It should be noted that the neural network-based approach has some drawbacks for application in software reliability assessment. The major problem is the design of neural network architecture, involving the number of input neurons in each layer and the number of hidden layers, so that both of them must be determined carefully through trial-and-error heuristics. Caruana [9] showed generalization results on a variety of problems as the size of the networks varies. Second, in related works on neural network application to predict the number of software faults, the prediction is always deterministic. In other words, the point prediction of the number of software faults detected at each testing day is given as an output of complex non-linear functions with trained parameters. In this case, it can be expected that the accuracy of future point forecast significantly reduces. More specifically, when the training data in neural networks is sparse, the point prediction in neural computing may be less reliable [4].

A feed forward neural network or multilayer perceptron (MLP) has an input layer, an output layer, and one or more hidden layers. Since a neural network with one hidden layer is capable of approximating any arbitrary function, in this paper we consider this simplest neural network architecture with only one single layer of hidden layer. One of the most important problems in neural network design is deciding the optimal size of the hidden layer (or the optimal number of hidden neurons) in order to achieve the best approximation/prediction performance for a given application. When the neural network is trained, the error of training set becomes small for training data. On the other hand, when new data is available to the neural network, the error may be extremely large [3]. In this paper, we concentrate on this issue and try to determine the optimal number of hidden neurons and input neuron through experiments.

In this paper we derive prediction intervals of the cumulative number of software faults detected at each testing day using the MLP neural network, where the simplest three layers MLP is assumed with the well-known back-propagation algorithm. To predict the number of software faults, we impose a plausible assumption that the underlying fault-detection process obeys the Poisson law with an unknown intensity parameter. Since it is appropriate to input the training data as real number in the conventional MLP neural networks, we propose to apply three data transform methods from the Poisson count data to the Gaussian data: Bartlett transform [5], Anscombe transform [1] and Fisz transform [12]. We experiment to find out the significance of the neural network architecture for input neuron and hidden neuron, and apply the delta method [4] to get the prediction intervals of the cumulative number of software faults in the on-going progress ahead prediction. In numerical experiments with actual software development project data, we show that our methods are useful to calculate the prediction intervals on the on-going progress of software debugging process.

2. Background

2.1 Neural Network Architecture

Artificial neural networks are widely used for functional approximation and statistical inference. The term of “artificial neural network” usually refers to mathematical model employed in neural network computing and artificial intelligence. Neural networks have the learning ability based on the training data or initial experiences, similar to the human brain. Although the neural network in the human brain is composed of a large number of highly interconnected processing elements (neurons) working in parallel, much simpler structure with input layer, hidden layer and output layer is assumed for the common MLP feed forward artificial neural network. It consists of an input layer with some inputs, hidden layer with hidden neurons and one output layer. The input layer of neuron can be used to capture the inputs from the outside world. The hidden layer of neurons has no communication with the external world, but the output layer of neuron sends the final output to the external world. The main function of hidden layer neurons is to receive the inputs and weights from the previous layer and to transfer the aggregated information to the output layer by any transfer function. This output can act as an input of the output layer. The input layer neurons do not have any computational task. It just receives inputs and associated weights, and passes them to the next layer.

The value coming out an input unit (neuron) is labeled by $x_i$ for $i = 1, 2, 3, ..., n$, where $n$ is the number of input neurons. There are also two special inputs, bias labeled by $B_1$ and $B_2$, which always have the unit values. These inputs are used to evaluate the bias to the hidden nodes and output nodes, respectively. Let $z_j (j = 1, 2, 3, ..., m)$ be the hidden neuron output, where $m$ is the number of hidden neurons in the hidden layer. Let $w_{ij}$ be the weight from $i$-th input to $j$-th hidden neuron, where $w_{0j}$ denotes the bias weight form $B_1$ to $j$-th hidden neuron. The bias input node $B_2$ is connected to all the hidden neurons and $B_2$ is connected to the output neuron. Figure 1 illustrates the MLP neural network architecture under consideration, where forward information...
corresponds to the data processing which is output from the feed forward architecture, and backward information denotes the data through the back-propagation algorithm. Each hidden neuron calculates the weighted sum of the input neuron, \( z_j \), in the following equation:

\[
z_j = \sum_{i=1}^{n} w_{ij} x_i + w_{0j} g_1 ,
\]

where

\[
g_1, g_2 = 1 \text{ for } B_1, B_2
\]

Substituting the weighted sum into the thresholding function which is typically either a step function or a sigmoid function, we can obtain the output from the hidden neuron \( \hat{z}_j \), where

\[
\varphi(z_j) = \hat{z}_j = \frac{1}{1+\exp(-z_j)}.
\]

Then, the output from the output layer is given by

\[
y = \sum_{j=1}^{m} w_{jd} \hat{z}_j + w_{0d} g_2,
\]

where \( y \) is the summative and weighted inputs from each hidden neuron of the output layer, \( w_{jd} \) is the weight going from the \( j \)-th hidden neuron to the output neuron, and \( w_{0d} \) represents the weight from the bias to the output neuron. Finally, the predicted value of the network, \( \hat{y} \), is calculated as

\[
\varphi(y) = \hat{y} = \frac{1}{1+\exp(-y)}.
\]

2.2 Back Propagation Algorithm

In our MLP architecture, all the processing units of input layer are interconnected to all the processing units of the hidden layers, and all the processing units of the hidden layer are interconnected to an output unit, where each weight is associated with each connection. The well-known gradient descent method is used to update the weights so as to minimize the squared error between the network output value and the target output value. Then, each weight is adjusted using the gradient descent, according to its contribution to the error. This procedure is iteratively made for each layer of the network, starting with the last set of weights, and working back towards the input layer, until the desired output is achieved. This is called the back propagation algorithm [27] and is the standard learning algorithm for artificial neural networks. The average sum of squared errors \( (E^2) \) for the input pattern is represented as

\[
E^2 = \frac{1}{N} \sum_{r=1}^{N} (Y_r - \hat{y}_r)^2 / (N - 1), \ r = 1, 2, ..., N,
\]

where \( \hat{y}_r \) is the predicted value (point prediction as the neural network output) at \( r \)-th testing day, \( Y_r \) is the true value of the number of detected software faults and \( N \) is the prediction period (integer value).

2.3 Data Transform

It is common to input real value data in the MLP neural network. Since our problem is to predict the number of software faults newly detected at the next testing day, however, the underlying data should be integer. In general, it is convenient to treat real number in almost all neural network computing, and to apply the useful property of the Gauss distribution for constructing the prediction intervals approximately (e.g. see [24]). Hence we suppose that the software fault count is described by the Poisson law [31][32][34][43]. In the existing literature, some of authors concern the prediction of the software fault-detection time and handle the real number in their neural network calculations.

We apply Bartlett transform [5], Anscombe transform [1] and Fisz transform [12] as the most well-known normalizing and variance-stabilizing data transforms. The Anscombe’s square-root transform (AT) is widely used to pre-process the Poisson data before processing the Gaussian data. Taking the AT, the cumulative number of software fault data can be approximately transformed to the Gaussian data:

\[
S_r = 2\sqrt{Y_r} + 3/8,
\]

where \( Y_r \) is the cumulative number of software faults at \( r \)-th testing day. The AT is a natural extension of the well-known Bartlett transform (BT), which is known as the most fundamental data transform tool in statistics, where BT is defined by \( B_r = 2\sqrt{Y_r} + 1/2 \). Finally, the Fisz transform (FT) is characterized by the following square root transform as an extension of BT:

\[
FT_r = \sqrt{Y_r + 1/4} + 1/2.
\]

2.4 Delta Method

Delta method is known as an elementary method of propagation of errors. It is a commonly used approach which is easily implemented, not computer-intensive, and can be robustly applied to many situations such as an approximation of the variance for an arbitrary functional of random variables, based on Taylor series expansions. It is also used to obtain an asymptotic statistical estimator from the knowledge of the limiting variance [17][24][41].
Let $\delta_r^T$ is the output gradient vector with respect to gradient values for all output and hidden neurons in the MPL neural network:

$$
\delta_r^T = [\delta_{O_1}, \delta_{O_2}, ..., \delta_{O_r}, \delta_{H_1}, \delta_{H_2}, \delta_{H_3}, ..., \delta_{H_m}],
$$

where $\delta_{O_r}$ ($r = 1, 2, 3, ..., N$) are the output gradient of the output layer, $\delta_{H_j}$ ($j = 1, 2, 3, ..., m$) are the output gradient of the hidden neuron, $m$ indicates the number of hidden neurons of a hidden layer and $a_r$ is the $r$-th actual value. Then, we have

$$
\delta_{O_r} = \hat{y}_r (1 - \hat{y}_r) (a_r - \hat{y}_r),
$$

$$
\delta_{H_j} = \delta_{O_r} w_{j,a} \varphi(H_j) (1 - \varphi(H_j)).
$$

In practice, the neural network parameters such as pervious weights have to be adjusted by minimizing the average $E^2$. Let $\Delta w_r$ is the Jacobian matrix with respect to all updated weight parameters from an output neuron to hidden neurons. It is computed for all the training samples, where

$$
\Delta w_r = 
\begin{bmatrix}
  w_{1d(new)} & w_{11(new)} & w_{21(new)} & ... & w_{11(new)} \\
  w_{2d(new)} & w_{12(new)} & w_{22(new)} & ... & w_{21(new)} \\
  w_{3d(new)} & w_{13(new)} & w_{23(new)} & ... & w_{31(new)} \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  w_{jd(new)} & w_{1j(new)} & w_{2j(new)} & ... & w_{ij(new)}
\end{bmatrix}.
$$

In the above equation, $w_{jd(new)}(d = 1)$ are the new weights of hidden neurons connected to the output neuron, and $w_{ij(new)}$ are the new weights of $n$ input neurons which are connected to $m$ hidden neurons in the hidden layer. These weights, $w_{jd(new)}$ and $w_{ij(new)}$, are given as follows.

$$
w_{ij(new)} = w_{ij} + \alpha w_{ij} + \eta \delta_{H_i} \hat{y}_r,
$$

$$
w_{jd(new)} = w_{jd} + \alpha w_{jd} + \eta \delta_{O_r} \hat{y}_r.
$$

Define the PIs of the cumulative number of software faults by $[P_{\text{low}}, P_{\text{upp}}]$ in the MLP neural network computing. Then, the lower limit and upper limit of PI are given by

$$
P_{\text{low}} = \hat{y}_r - t_{N,p}^{1/2} E \sqrt{1 + \delta_r^T (\Delta w_r^T \Delta w_r)^{-1} \delta_r},
$$

$$
P_{\text{upp}} = \hat{y}_r + t_{N,p}^{1/2} E \sqrt{1 + \delta_r^T (\Delta w_r^T \Delta w_r)^{-1} \delta_r}.
$$

In the above equations $t_{N,p}^{1/2}$ is the $(\alpha/2)$ - quantile of the student $t$-distribution function with $(N-p)$ degree of freedom, and $p$ is the number of inputs in the neural network [24].

The Jacobian matrix $(\Delta w_r)$ and its gradient value $\left( \delta_r^T \right)$ are quite hard to obtain with all input data, so delta method contains a somewhat puzzling issue to construct PIs. However the other calculations are comparatively modest. In this paper, the Jacobian matrix and the gradient value are calculated and estimated at off-line, although they can be potential sources of computational error for constructing PIs. In addition, the quality of PIs and their optimal values of gradient and Jacobian matrix must be carefully checked to satisfy the convergence condition that the minimum error is achieved at a tolerance level.

3. Numerical Illustrations

3.1 Experimental process

The overall experimental process is described in Figure 2. In the first step, we consider two cases on preprocessing the dataset; one is data transforming from the Poisson data to the Gaussian data with BT, AT and FT, another without transform. The data with/without transform are input to the neural network, where the output are the point prediction of the cumulative number of software faults detected at the next testing day. At the last step we assess the PIs of the cumulative number of software faults by using the delta method.

3.2 Setup

We use eight real project data sets cited in [31]; DS1~DS8, which consist of the software-fault count data. Table I summarizes the data sets and their cumulative numbers of software faults detected in testing. To find out the desired output via the back propagation algorithm, we need much computation cost to calculate the gradient descent and the Jacobian matrix. Especially, the momentum $(\alpha)$ and the learning rate $(\eta)$ are the most important turning parameters, where $\alpha$ adjusts the weights and $\eta$ depends on the convergence speed in the back propagation algorithm. We carefully examine $\alpha$, $\eta$, the initial guess of weight, number of total iterations and the tolerance level parameters in pre-experiments [4].
3.3 Prediction Performance

<table>
<thead>
<tr>
<th>DS #</th>
<th>Number of Faults</th>
<th>Project Type</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>62</td>
<td>Command and Control subsystem</td>
<td>Data &amp; Analysis Center for Software (DACS) [31]</td>
</tr>
<tr>
<td>2</td>
<td>41</td>
<td>Flight Data subsystem</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>114</td>
<td>Command and Data subsystem</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>22</td>
<td>Real Time Command &amp; Control</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>73</td>
<td>Commercial Subsystem</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>181</td>
<td>Command and Data subsystem</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>81</td>
<td>Brazilian Electronic Switching System</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>140</td>
<td>Telecommunications switch software</td>
<td></td>
</tr>
</tbody>
</table>

The prediction performance is evaluated in sequential software testing, so that we make the one-stage look ahead prediction based on the past observation and sequentially evaluate the prediction performance. As a prediction performance measure, we introduce the average relative error. Suppose that the observation point is the $k$-th testing day and that $k$ software fault counts data are available. For the actual value on the number of software fault counts at the $k$-th testing day, the relative error ($RE_k$) is defined by

$$RE_k = \left| \hat{Y}_k - Y_k \right| / Y_k,$$

where $\hat{Y}_k$ is the prediction value at $k$-th testing time and $Y_k$ is the $k$-th actual value (training data). The average relative error takes account of the past history and is defined by $AE = \frac{1}{N} \sum_{k=1}^{N} RE_k$.

A Effect of number of hidden nodes

One of the major problems facing in software fault prediction is the selection of input and hidden neurons number for the architecture of neural networks. To keep relatively small errors on software fault prediction, it is very important to train the neural network. The minimal error reflects better stability, and higher error reflects worst stability. We experiment with different numbers of hidden neurons for a fixed number of input neurons for all datasets. The excessive hidden neurons cause the so-called over fitting problem and tend to overestimate the complexity of the target problem. To choose the optimal number of hidden neurons we follow the well-known "rules of thumb" for choosing a suitable architecture. In this paper we determine the number of hidden nodes by $(2/3)(\text{number of inputs}+\text{outputs})$ [44].

Table II lists abbreviations used hereafter.

B Effect of number of input nodes

In most situations, there is no way to determine the best number of input nodes without training several networks and estimating the generalization error. If we have too many input neurons, we may get higher training error and higher generalization error, due to under-fitting and large statistical bias. Once the best number of hidden neurons is known, we can change the input neuron numbers and can make a
different architecture to find out the optimal number of input neurons. Table IV presents the prediction performance based on AE for all datasets with and without data transform. It is shown that when the number of input neurons increases, the error rate also increases, but some cases e.g., DS#3–DS#5 without transform provide the better results for 11:7:1, 12:10:1 and 12:7:1 than FT. It should be noted that the neural network has one major drawback for application in software reliability assessment, i.e., in the common neural network computing the initial weight is randomly selected, so it acts as a trial-and-error heuristic.

Table II. Terminology.

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN</td>
<td>Neural Network</td>
</tr>
<tr>
<td>PI</td>
<td>Prediction Interval</td>
</tr>
<tr>
<td>MLP</td>
<td>Multilayer Perceptron</td>
</tr>
<tr>
<td>ANN</td>
<td>Artificial Neural Network</td>
</tr>
<tr>
<td>SRGM</td>
<td>Software Reliability Growth Model</td>
</tr>
<tr>
<td>PICP</td>
<td>PI coverage Probability</td>
</tr>
<tr>
<td>MPIW</td>
<td>Mean Prediction Interval Width</td>
</tr>
<tr>
<td>PINAW</td>
<td>PI Normalized Average Width</td>
</tr>
<tr>
<td>RE</td>
<td>Relative Error</td>
</tr>
<tr>
<td>AE</td>
<td>Average Relative Error</td>
</tr>
</tbody>
</table>

Table III. Average Relative Error in Point Prediction.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Average Relative Error</th>
<th>Architecture</th>
<th>AT</th>
<th>FT</th>
<th>BT</th>
<th>No Transform</th>
</tr>
</thead>
<tbody>
<tr>
<td># of DS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>5:3:1</td>
<td>0.3370</td>
<td>0.1041</td>
<td>0.4139</td>
<td>0.1433</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5:4:1</td>
<td>0.5148</td>
<td>0.9771</td>
<td>1.3048</td>
<td>2.1271</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5:5:1</td>
<td>0.5236</td>
<td>0.1253</td>
<td>0.4253</td>
<td>0.3215</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5:3:1</td>
<td>0.2417</td>
<td>0.5412</td>
<td>1.1774</td>
<td>1.1478</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5:4:1</td>
<td>0.1639</td>
<td>0.3680</td>
<td>0.8222</td>
<td>0.7548</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5:5:1</td>
<td>1.5236</td>
<td>1.0235</td>
<td>3.1725</td>
<td>1.9356</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>10:5:1</td>
<td>1.7969</td>
<td>0.5747</td>
<td>0.9225</td>
<td>0.9162</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10:7:1</td>
<td>0.9043</td>
<td>0.2685</td>
<td>0.7069</td>
<td>0.6712</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10:8:1</td>
<td>2.5369</td>
<td>1.2536</td>
<td>3.1235</td>
<td>3.3698</td>
<td></td>
</tr>
<tr>
<td></td>
<td>14:5:1</td>
<td>1.8627</td>
<td>1.2515</td>
<td>1.9189</td>
<td>1.3079</td>
<td></td>
</tr>
<tr>
<td></td>
<td>15:10:1</td>
<td>0.6845</td>
<td>0.1222</td>
<td>0.8647</td>
<td>0.7851</td>
<td></td>
</tr>
<tr>
<td></td>
<td>15:15:1</td>
<td>1.1627</td>
<td>1.0488</td>
<td>2.1147</td>
<td>2.3878</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5:5:1</td>
<td>1.3895</td>
<td>0.9713</td>
<td>1.1568</td>
<td>0.6985</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10:7:1</td>
<td>0.8704</td>
<td>0.7001</td>
<td>0.9881</td>
<td>0.9257</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10:8:1</td>
<td>1.7845</td>
<td>1.0134</td>
<td>1.1575</td>
<td>1.6134</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>20:5:1</td>
<td>2.1763</td>
<td>0.9861</td>
<td>0.5412</td>
<td>1.1398</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20:10:1</td>
<td>1.1368</td>
<td>0.9763</td>
<td>0.6594</td>
<td>2.7634</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20:14:1</td>
<td>0.9852</td>
<td>1.1207</td>
<td>0.7613</td>
<td>1.2078</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>10:5:1</td>
<td>1.4713</td>
<td>0.9231</td>
<td>1.4801</td>
<td>0.9474</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10:7:1</td>
<td>0.9958</td>
<td>0.2487</td>
<td>1.2128</td>
<td>0.7487</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10:8:1</td>
<td>0.7856</td>
<td>0.0945</td>
<td>0.9645</td>
<td>1.2356</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5:3:1</td>
<td>1.9457</td>
<td>1.3531</td>
<td>1.8557</td>
<td>0.9147</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5:4:1</td>
<td>0.8387</td>
<td>0.6914</td>
<td>1.3731</td>
<td>0.9537</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5:5:1</td>
<td>2.0931</td>
<td>1.0537</td>
<td>1.4526</td>
<td>1.0535</td>
<td></td>
</tr>
</tbody>
</table>

3.4 One-stage look ahead point prediction

To get the one-stage look ahead point prediction value, we use DS # 4. Noting that our dataset contains 22 days, we wish to know 23th point prediction value of the cumulative number of software faults for all transform methods. From the result of Tables III and IV, it is seen that FT can give less error rate and more reliable prediction results than the others. Figure 4 shows that result based on the best neural network architecture. From this figure we can say that FT gives little larger value than the others.

3.5 PI Assessment

In order to measure the quality of PIs on the prediction of software fault counts, we need to define three prediction measures called the PI coverage probability (PICP), the mean prediction interval width (MPIW) [4] and PI-normalized averaged width (PINAW) [25]. Assume the significance level as 95%. PCIP is the portion of the number of the software fault counts covered by the PI, and is defined by $\text{PICP} = \frac{\sum_{i=1}^{N} I(C_i)}{N}$.
where
\[ c_i = \begin{cases} 1, & \hat{y}_i \in [L_i, U_i], \\ 0, & \hat{y}_i \notin [L_i, U_i]. \end{cases} \]

Let \( L_i \) and \( U_i \) be the lower and upper prediction limits, \( CP_i \) be the coverage probability at \( i \)-th testing day (\( i = 1, 2, \ldots, k \)) and \( \hat{y}_i \) be the predicted number of software faults. Then, \( MPIW \) evaluates the width of PIs, and is defined by
\[
MPIW = \frac{\sum_{i=1}^{N} (U_i - L_i)}{N}.
\]

PI-normalized averaged width (PINAW) quantifies the wide constructed PIs; \( PINAW = \frac{MPIW}{R} \), where \( R \) is the range of the underlying target, and is used to compare PIs.

In Table V we give the prediction PI measures for eight datasets. It can be seen that the no transform works to increase PICP because the corresponding MPIWs become wider. In practical usage of PIs, the associated coverage rate is large enough if the width is narrow. From this table we can say that in most of the cases FT provides narrow width with higher coverage rate excluding DS# 6. On the other hand, BT can give the better result than FT. Since the sharper PIs are theoretically more informative and practically more useful than the wider PIs, it is noted that PINAWs indicate the sharpness of PIs, so that smaller PINAW is regarded as better PIs.

Since software is written by humans, errors will be always involved in the product. From this well-known fact, it can be recognized that the PIs can predict the number of software fault counts under uncertainty, which will experience in the future operational phase, and can be useful for the probabilistic inference with subjective significance level controlled by the software test manager.
In Figs. 5-8, we depict the sequential prediction results of software fault counts and their PIs with four data transform methods (including the case with no transform) for the different datasets, where the best architecture in each dataset is applied. It can be found that the PIs with FT can cover both of the one-stage look ahead point prediction and the actual data itself. Figure 7 shows that BT for DS#7 does not cover the point prediction in PIs. Figure 8 illustrates that the case with no transform for DS#8 contains the PI coverage.

4. Discussion and Conclusion

In this paper we have derived the prediction interval of the cumulative number of software faults in testing phase and investigated the effect of a number of hidden and input nodes on prediction accuracy according to the rules of thumb. The experimental results have shown that the proposed approach gave the acceptable results for prediction using the different neural network architectures. In numerical experiments with actual software development project data, we have evaluated the resulting prediction intervals and found that those could cover both the point prediction and the actual data in their regions.

In future, these experimental results have to be justified through Monte Carlo simulation, by comparing the “real” prediction intervals under the well-defined parametric circumstance. Also, we will apply the other PIs methods to construct the long-term prediction and optimize the weights for the neural network architecture to get the exact PIs.

Acknowledgement

The first version of this paper was presented at The 3rd International Conference on Applied Computing & Information Technology (ACIT2015), Okayama, Japan, July 12-16, 2015. The first author was fully supported by the MEXT (Ministry of Education, Culture, Sports, Science, and Technology), Japan government scholarship.

References


[29] T. M. Khoshgoftaar, “Using Neural Networks to Predict Software Faults During Testing,” IEEE


Begum Momotaz received the B.Sc. and M.Sc. degrees in Computer Science & engineering from Dhaka University of Engineering & Technology (DUET), Gazipur, Bangladesh in 2008 and 2013, respectively.

In 2013, she joined the Department of Computer Science & Engineering, DUET, Gazipur, Bangladesh, as an Assistant Professor. She is now working toward the Ph.D. degree in the Graduate School of Engineering, Hiroshima University, Japan.

Tadashi Dohi received the B.Sc., M.Sc., and Ph.D. degrees in engineering from Hiroshima University, Hiroshima, Japan, in 1989, 1991, and 1995, respectively. In 1992, he joined the Department of Industrial and Systems Engineering, Hiroshima University, Japan, as an Assistant Professor.

Since 2002, he has been working as a Full Professor in the Department of Information Engineering, Graduate School of Engineering, Hiroshima University. In 1992 and 2000, he was a Visiting Research Scholar in University of British Columbia, Canada, and Duke University, Durham, NC, USA, respectively, on leave of absence from Hiroshima University. His research areas include software reliability engineering, dependable computing, and performance evaluation. He is a regular member of ORSJ, IEICE, IPSJ, REAJ and IEEE.
An Empirical Study on Recovering
Requirement-to-Code Links

Zhang Yuchen*, Wan Chengcheng*, Jin Bo†
*School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China
†The Third Research Institute of Ministry of Public Security, Shanghai, China
zhangyc005@sjtu.edu.cn, 2638807600@qq.com, jinbo@stars.org.cn

Abstract—Requirements traceability provides support for
critical software engineering activities such as change impact
analysis and requirements validation. Unfortunately many
organizations have ineffective traceability practices in place,
largely because of poor communication and time pressure
problems. Therefore researchers have proposed various
approaches to automatically recover requirement-to-code links.
Typically, these approaches are based on Information Retrieval
techniques, and use various features such as synonyms,
verb-object phrases, and structural information. Although many
links are thus recovered, the effectiveness of individual features
is not fully evaluated, and it is rather difficult to combine
different features to produce better results. In this paper, we
implement a tool, called R2C, that combines various features to
recover requirement-to-code links. With the support of R2C, we
conduct an empirical study to understand the effectiveness of
these features in recovering requirement-to-code links. Our
results show that verb-object phrase is the most effective
feature in recovering such links. A preliminary case study
indicates that our tuning combines different features to produce
better results than IR-based technique using a single feature.

Keywords—Requirement-to-Code Links, Traceability Recovery,
Verb-Object Phrase, Information Retrieval

I. INTRODUCTION

Software traceability is recognized as an important quality
of a well-engineered software system [1]. The traceability
information, including links between requirements and code,
is critical in the management of software development [2], so
many companies carefully define such traceability
information before development. In practice, some companies
may not define the traceability carefully, and even if such
traceability initially is defined, it soon becomes obsolete,
because both requirements and implementations are changing
due to various factors [3]. As a result, programmers often
have to maintain the links between requirements and code
manually, which is time-consuming and error prone [4].

To reduce the heavy manual effort for maintaining such
links, automatically recovering traceability has long been a
hot research topic in the software engineering
community [5]. Researchers have proposed many approaches,
and most of them are based on information retrieval (referred
to as IR-based approaches in this paper). Typically, IR-based
approaches extract terms to represent requirements and code,
and compare extracted terms to build the links between
requirements and code. Although IR-based approaches
successfully recover many links, their precisions and recalls
are often below expectation.

We manually inspected many requirements and their
corresponding code snippets and find that the following three
features are widely used. First, although requirements and
code use different terms, these terms are often synonyms.
Second, requirements often contain many irrelevant words,
whereas verb-object phrases convey the essential meanings.
Third, code that linked to the same requirement always have
relationships, such as function calls, inheritance or realization
relationships. About the effectiveness of these different
features, many research questions in this research direction
are still open. For example, which is the most effective
feature to recover the links? How to integrate different
features for the best results?

In this paper, we conduct an empirical study to address
the above research questions. This paper makes the following
major contributions:

- We implement a tool, called R2C, that recovers links
  between requirements and code. With the support of
  R2C, we conduct an evaluation on four real projects.
  Our results show that R2C is advanced, since it
  achieves even better results than a state-of-the-art
tool.

- Empirically, we show that verb-object phrases are
  the most important feature to recover the links
  between requirements and code, and combining
  various features can lead to better results than
  individual features.

The rest of this paper is structured as follows. Section II
introduces semantic features in existing literatures.
Section III presents our empirical study. Section IV
introduces related work. Section V concludes.

II. FEATURES

In this section, we present three key features in
literatures. Different approaches implement similarity
formulæ with subtle differences. It is infeasible to implement
all the formulæ, but we implement reasonable formulæ in
R2C.

A. Three Key Features

We find that most approaches use the following features:

- **Synonyms.** IR-based approaches extract terms from
  requirements and code. Although these terms are not
identical, they are often synonyms, when a requirement and a piece of piece are linked correctly.

- **Verb-object phrases.** In requirements, most sentences have verb-object phrases, which convey the essential meanings of sentences. We can catch the main information of the requirements with the support of natural language processing (NLP) techniques.

- **Structural Information.** There are lots of structural information potentially contained in source code. In traceability links, source code related to the same requirement always have relationships.

### B. Synonym

Although synonyms are important, it is nontrivial to determine synonyms of terms. Some recent approaches [6], [7] define their own glossary and ontology, and their limitation lies in the huge manual effort to build such glossary or ontology. To reduce the human effort, R2C uses WordNet [8] to locate synonyms of a given term. In WordNet, each term can have several senses, and each sense is stored as a semantic tree. The leaves of a semantic tree are words with similar meanings. There are many researches in calculating the similarity between two senses of terms, a common method is based on the synset, classword and sense explanation in WordNet [9]. Given two terms \( S_1 \) and \( S_2 \), their approach [9] calculates their similarity as \( \text{sim}(S_1, S_2) \). However, a term may have different senses, and it is difficult to determine its sense in a context. We find that in many pairs of terms, their correct senses have the maximum values of \( \text{sim}(S_1, S_2) \). As a result, we define the similarity between two terms as follows:

\[
WSim(W_1, W_2) = \max \{ \text{sim}(S_i, S_j) \} \tag{1}
\]

where \( S_i \) represents a sense of word \( W_1 \), and \( S_j \) represents a sense of word \( W_2 \).

### C. Verb-object Phrases

In traditional IR techniques, a document is regarded as a bag of unordered words. Typically, these techniques focus on only nouns, since they are designed to understand the key idea of a document. In such cases, nouns are often more important than the other words.

However, the assumption does not fit the context of recovering links between requirements and code. In software projects, nouns are often limited, so many nouns in arbitrary requirements and code snippets are identical. As a result, it leads to many errors, when traditional approaches try to recover links based on only nouns.

We find that in many incorrect requirement-to-code links, although their nouns are identical, their verbs are different, since they describe actions. For example, we investigated the requirements and code of a hospital information management system, and we find that both documents and code contain some nouns (e.g., drug) many times. These words are often not helpful in recovering the requirement-to-code links. In the contrast, from the perspective of actions, requirements and code are distinct, since they are related to different actions (e.g., buying drugs). As a result, it may achieve better results, if we extract verb-object phrases from requirements and code comments.

R2C uses Stanford Parser [10] to analyze syntactic structure of sentences. Based on the built syntax trees and Part-of-speech tags, R2C extracts the verb-object phrases from both requirements and code comments. Extracted verbs and nouns may be in morphological forms. R2C reduces these words to its root word with the stemmer of Stanford Parser. For example, it reduces book and books to the root word, book.

As code comments are different from code elements, R2C treats them in different ways:

### Requirements & Code comments.** Both requirements and code comments are in natural language, R2C extracts terms from them with the same following steps:

- **Part-of-speech (POS) tagging.** R2C first builds POS tags for each sentence of a given code comments. These POS tags include verbs, nouns, adjectives, adverbs, and others.

- **Parsing.** R2C analyzes syntactic structure of sentences and the dependency between terms.

- **Extracting verb-object phrases.** R2C extracts a verb-object phrase from each simple sentence or each clause of a complicated sentence. Extracted verb-object phrases are in the form of \( \langle \text{verb}, \text{noun} \rangle \).

- **Stemming verb-object of phrases.** R2C reduces verbs and objects to their root words.

### Code elements.** Code elements are in programming languages. R2C extract verb-object phrases from code elements with the following steps:

- **Extracting identifiers.** R2C extracts identifiers such as class names, method names and variable names from a given piece of code.

- **Extracting words.** R2C splits names of identifiers into sets of words. For example, \( \text{getUserInfo} \) is split into \( \{ \text{get}, \text{user}, \text{info} \} \), and \( \text{system.initialize} \) is split into \( \{ \text{initialize}, \text{system} \} \).

- **Recovering acronyms.** R2C extends minimum edit distance algorithm [11] to compare the distances from words in code comments and an acronym, and resolves the acronym as the word with minimum distance. For example, \( \text{info} \) is resolved as \( \text{information} \).

- **Extracting and stemming verb-object phrases.** R2C combines words into simple sentences. After that, it extracts and stems verb-object phrases as it extracts such phrases from code comments.

We find that verb-object phrases between requirements and code may not be identical. For two given verb-object phrases \( \langle V_1, N_1 \rangle \) and \( \langle V_2, N_2 \rangle \), R2C defines their similarity as follows:

\[
PSim(P_1, P_2) = \alpha \times WSim(V_1, V_2) + (1 - \alpha) \times WSim(N_1, N_2) \tag{2}
\]
where \( W Sim(W_i, W_j) \) is defined in Equation 1; \( \alpha \) and \( (1-\alpha) \) are the weights of verbs and nouns, respectively.

D. Computing Text Similarity

Following other state-of-the-art tools, R2C treats requirements and code as texts and compare their similarities based on IR techniques.

To compute the similarity between requirements and code, R2C adopts Vector Space Model (VSM), a widely used model in IR. In VSM, a document is represented as an \( n \)-dimensional vector \( < w_1, w_2, \cdots, w_n > \), where \( n \) represents the number of distinct terms such as words or phrases, and \( w_i (1 \leq i \leq n) \) represents the weight of a unique term. R2C calculates the weight for each term based on Frequency-Inverse Document Frequency (TF-IDF) metric. In TF-IDF, the weight \( w_i \) of a term in a document increases with its occurrence frequency in this document and decreases with its occurrence frequency in all documents. It is defined as follows:

\[
 w_i = tf_i \times \log \left( \frac{|D|}{|\{j : t_i \in d_j\}|} \right) \tag{3}
\]

where \( tf_i \) represents the occurrence frequency of term \( t_i \) in a document, \( |D| \) represents the number of documents, and \( \{|j : t_i \in d_j\}| \) represents the number of documents that contain the term \( t_i \). It is noteworthy that we calculate TF-IDF values of terms for requirements and code separately.

The TF-IDF metric is designed for terms in natural language documents. Code is in programming languages, which are quite different from natural languages. First, class names play more important roles in expressing the functionality of code than variable names. Second, methods with more lines of code are more important than those methods with only several lines of code. Based on the above findings, we use the parameter \( \eta \) to tune the weights \( w_i \) of terms as follows:

\[
 \eta = \gamma \times \log(LOC) \tag{4}
\]

where \( \gamma \) represents the importance of these terms, and LOC represents lines of code. If terms are not extracted from code, we set \( \eta \) as 1. In addition, as discussed before, terms may not be identical but synonyms. Considering this, we tune the weight \( w_i \) as follows:

\[
 w_i = \sum_{\{j : (P_i, P_j) \in S\}} (\eta \times tf_i \times idf_i \times PSim(P_i, P_j)) \tag{5}
\]

where \( \{j : (P_i, P_j) \in S\} \) represents iterating all synonymous terms \( P_j \) of a term \( P_i \).

After R2C builds VSM based on TF-IDF, it defines the similarity between requirements and code as the cosine of the angle between the corresponding vectors.

\[
 Sim(r, c) = \frac{V_r \times V_c}{\|V_r\| \times \|V_c\|} \tag{6}
\]

where \( V_r \) and \( V_c \) are the vectors that denote requirements and code, and \( \|V\| \) represents the Euclidean norm of vector \( V \).

Algorithm 1: Updating with structural information

E. Structural Information

Consider an example in a small book management system. There is a requirement \texttt{AuthorizeUser} and a source code class \texttt{User.java}. With IR-based approach mentioned before, it is easy to recover a link between them. However, there are some other correct links missed. The requirement also relevant to \texttt{Reader.java} which extends \texttt{User.java}. Injecting structural information, R2C is able to recover this missed link because of the inheritance relationship between \texttt{User.java} and \texttt{Reader.java}. After recovering a set of initial links, R2C next uses structural information to update similarity values between requirements and code.

In Algorithm 1, \( G(C, E) \) is the indirect graph of relationships between source code, in which \( C = \{c_1, c_2, \cdots, c_n\} \) means the set of code and \( E = \{(c_i, c_j)\} \) means the set of relationships. The relationships include call relations and inheritance relations. Moreover, let \( S \) be the set of requirement documents and \( Links \) be the set of initial links recovered in the previous step. \( Sim(s,c) \) is the similarity calculated with Equation 6. If a requirement is linked to a code, it is probably linked to some other related code, so we increase their similarity value by adding an extra \( \delta \times Sim(s_i, c_j) \). Although adding the extra values improves the similarity calculation, it is difficult to determine \( \delta \). Considering the size can sensibly differ from one system to another, we set a adaptive \( \delta \) as follows [12]:

\[
 \delta = \text{medium} \left\{ (\max_i - \min_i)/2 \right\} \tag{7}
\]

where \( \max_i \) and \( \min_i \) are the maximum and minimum similarity values between all requirement documents and source code, and \( \text{medium} \) denotes the variability.

III. EMPIRICAL STUDY

In this study, we focus on two following research questions:

- To what degree does our approach combine various features to produce better results (Section III-C)?
- Which is the most effective feature to recover requirement-to-code links (Section III-D)?

To answer the first research question, we compare R2C with an IR-based tool (baseline) [13] and two improved tool phrasing [14] and O-CSTI [12]. The results show that R2C is an advanced tool, which increases the reliability of the results in the follow-up research question. To answer the second research question, we combine different features. Our results show that verb-object phrases are the most effective feature to recover requirement-to-code links.
A. Setup

Table I shows the subject projects in our study. The requirements and code comments of all these projects are in English, but some nouns in eTour and EasyClinic are in Italian. In eTour, iBooks and SMS, most classes, methods have code comments and some important code lines have code comments as well. In EasyClinic, only classes have code comments.

In our study, we need the ground truth of links between requirements and code. SMS provides the links between its requirements and code, so we use these links as the golden standard. The other three projects do not provide such links, so we build them by experts. Comparing the manual results with our results, we calculate the recalls, precisions and F-measures of R2C.

B. Metrics

A recovered link falls into one of the four categories, i.e., a link that is identified as a true link (TP), a link that is identified as a false link (FP), a true link that is missed (TN), and a false link that is missed (FN). Based on these categories, we define recall, precision and F-measure as follows:

\[
Recall = \frac{TP}{TP + FN} \quad (8)
\]

\[
Precision = \frac{TP}{TP + FP} \quad (9)
\]

\[
F-measure = \frac{2 \times Recall \times Precision}{Recall + Precision} \quad (10)
\]

C. Overall Results

Table II shows the overall result. In the eTour project, the golden standard has 366 links between 58 requirements and 116 methods. The baseline approach recovered 186 (50.82%) valid links from 402 (46.27%) identified links. Phrasing and O-CSTI are better, respectively recovered 176 (48.09%) from 349 (50.43%) and 188 (51.37%) from 371 (50.67%). The R2C performed the best, recovered 197 (53.83%) valid links from 312 (63.14%) identified links. As F-measures are concerned, R2C outperforms the baseline approach from 0.4844 to 0.6170. In the iBooks project, the golden standard has 104 links between 19 requirements and 61 methods. R2C recovered 97 (93.62%) of them, with the precision 77.17%. Compared with the baseline approach, Phrasing and O-CSTI, both recall and precision in R2C are significantly higher, which leads to a better F-measure. The SMS project is largest project in our study, with 1071 links between 64 requirements and 102 methods. R2C recovered 638 (59.57%) valid links, while the baseline approach, Phrasing and O-CSTI respectively recovered 597 (55.74%), 602 (56.21%) and 591 (55.18%) valid links. In total, the other three approaches recovered 1462, 1317 and 1380 links, whereas R2C identified only 1049 links. R2C improves the precision as well, leading to a better F-measure with 0.5925. In the EasyClinic project, the golden standard has 93 links between 30 requirements and 47 methods. In both Recall and Precision, R2C is marginally higher (less than 7%) than the other three approaches. Table II shows the overall result when F-measure achieve the maximum.

As industrial applications of automated traceability are only considered successful when they achieve high recall levels [15], we also evaluate the precision of all four projects at a fixed recall level close to 0.9, and the result is shown in Figure 1.

In summary, our results show that R2C is more effective than the baseline approach. The results show that R2C is an advanced tool, and increases the reliability when we evaluate the most effective feature.

D. The Most Effective Feature

To evaluate the effectiveness of individual features, we rerun our evaluation with different settings. In particular, based on the baseline approach, R2C-a denotes the setting when we use only synonym feature; R2C-b denotes the setting when we use only verb-object phrase feature; and R2C-c denotes the setting when we use only structural information feature. Then we also combines these features to understand their impacts on each other.

We select the iBooks project in this study, and Figure 2 shows the results. As shown in Figure 2, the combination of arbitrary two features improve the results of a single feature, and the combination of all the three features improve the results of any two features. The result highlights our tuning algorithm, since it combines features to produce better results. As far as a single feature is concerned, our results
show that the verb-object phrase is the most effective feature to recover requirement-to-code links.

E. Discussion

Our results show that R2C is more effective in recovering requirement-to-code links than the other approaches, since it achieves higher F-measure 0.07 - 0.20, or higher precision 5% - 23% at a fixed recall. We find that the higher F-measure comes from more accurate extracted terms and structural information. All these four approaches are based on VSM, the baseline approach and O-CSTI extract keywords by only removing stop words, whereas R2C and Phrasing extract phrases to accurately convey the major meanings of documents without losing much information. With the initial links R2C and O-CSTI use structural information of source code to improve the result, while the baseline approach and Phrasing did not. Overall R2C integrate many techniques (e.g., source code identifier processing, semantic analysis, synonym identification and source code structural analysis) to achieve the best result.

However, R2C still fail to recover some links. We find four issues for further improvements. First, as we use only a verb-object phrase to represent a sentence, it may lose information in some cases. Second, we use WordNet to find synonyms of terms. As WordNet does not include all terms, we fail to identify synonyms for some terms. Third, some programmers do not write code comments or name their identifiers meaningfully, which reduces the effectiveness of our approach. For example, we notice that sometimes programmers write their private affairs in code comments, and such comments are extracted as terms by our approach. Last, we only use function calls, inheritance or realization relationships from source code structural analysis, there may be some other useful structural information we ignored.

In our study of overall result, we find the improvement in the EasyClinic project is not so distinct as in other projects. We checked the project, and find two factors. One is that in this project, names of identifiers are in Italian, so we fail to extract proper verbs and nouns from code. The other factor is that in this project, programmers write comments for only classes, so extracted terms are insufficient to recover links.

Why verb-object phrases are so important in recovering requirement-to-code links? We manually check the documents of iBooks project and find the reasons. There is a sentence "Create an initial information of borrowing books" in UC5, and "User login, and create a request of borrowing books" in UC14. Without verb-object phrases extraction, this sentence in UC5 is represented as the vector \{create, user, initial, information, borrow, book\} and the sentence in UC14 is represented as \{user, login, create, request, borrow, book\}. As VSM is unordered, these two vectors are so similar that both UC5 and UC14 may be linked to User.java that contains a method create(). In fact, only UC5 should be linked to User.java and UC14 has no relation with this code file. When we use verb-object phrases as the terms of VSM, the key information of these sentences are represented as follows: \{(create, user), (initial, information)\} for UC5 and \{(login, user), (create, request), (borrow, book)\} for UC14. Using verb-object phrases as the terms of VSM, the similarity between UC5 and User.java still return a high value, but that between UC14 and User.java return a low value with phrases mismatching. Then we can obviously identify the valid link and irrelevant link by comparing their similarity values with the threshold.

IV. RELATED WORK

Most approaches in literature are based on information retrieval (IR) algorithms, such as vector space model (VSM) [16] and probabilistic network model (PNM) [13] [17], which calculate the calculate of links based on the frequency and distribution of terms. For example, Hayes et al. [16] conjunct a thesaurus with the VSM to establish the links between requirements and code. LSI [18] and Latent Dirichlet Allocation (LDA) [19] [20] were applied to understand the semantics context of terms in the requirements and code from the viewpoint of probability, which is not real understand the meaning of terms. To real understand the semantics context of terms from natural language, ontology and glossaries [6] [7] were merged into existing approaches.

There are also some researchers combine textual and structural analysis of software artifacts for traceability link recovery [12]. Based on existing traceability links recovered by traditional IR-based approaches or manual search, these tools recovered more missing links by using the structural information of both documentations and source code. To improve the precision of result, researchers try some self-adapts approaches to learn from human feedback. A limited number of tools integrate user feedback. For example, ADAMS [18], DOBRET [21] and RETRO [19]
collect relevance feedback on the links which have been 
created automatically using IR techniques. Users can increase 
or decrease term weighting used to compute the similarity 
according to whether a term occurs in a rejected or confirmed 
link [16] [22], and they also use eye-trackers to explore how 
project analysts verify links between requirements and code. 
However, this kind of approaches is still not good enough, 
and these information are the key for the performance 

improvement in recovering links.

As an improvement of the existing approaches, we 
proposed a integrated approach to recover links 
requirement-to-code traceability links (R2C). R2C applied 
some new techniques (i.e. semantic analysis, and structural 
analysis etc.) on three widely important features, then 
achieved more information from requirements and code. And 
the quality of the starting matrix, the worse the decision the analysts make [24].

V. CONCLUSION AND FUTURE WORK

This paper presents an approach R2C to recover 
traceability links between requirements and code. With the 
support of R2C, we conduct an empirical study on three 
features. Results show that R2C combines various features to 
produce better results, and the most effective feature is the 
verb-object phrase in recovering links.

It is worthy mentioning that several issues need further 

studies. First, R2C analyzes requirements only in for English, 
we plan to improve and apply it in other languages. Second, 
there are still much space for improving the precisions and 
calls of our approach. Future work will be devoted to using 
global optimization techniques based on feedback, to further 
perform better results, and to produce better results, and the most effective feature is the 
verb-object phrase in recovering links.

It is worthy mentioning that several issues need further 

studies. First, R2C analyzes requirements only in for English, 
we plan to improve and apply it in other languages. Second, 
there are still much space for improving the precisions and 
calls of our approach. Future work will be devoted to using 
global optimization techniques based on feedback, to further 
perform better results, and to produce better results, and the most effective feature is the 
verb-object phrase in recovering links.

It is worthy mentioning that several issues need further 

studies. First, R2C analyzes requirements only in for English, 
we plan to improve and apply it in other languages. Second, 
there are still much space for improving the precisions and 
calls of our approach. Future work will be devoted to using 
global optimization techniques based on feedback, to further 
perform better results, and to produce better results, and the most effective feature is the 
verb-object phrase in recovering links.

ACKNOWLEDGMENT

This research is supported by 973 Program in China 
(Grant No. 2015CB352203) , National Natural Science 
Foundation of China (Grant No. 61472242), and Key Lab of 
Information Network Security, Ministry of Public Security 
(Grant No. C14609). Jin Bo is the corresponding author.

REFERENCES

[1] Orlena CZ Gotel and Anthony CW Finkelstein. An analysis of the 

[2] Patrick M’der and Alexander Egyed. Do developers benefit from 
requirements traceability when evolving and maintaining a software 


[4] Jane Huffman Hayes, Alex Dekhtyar, and James Osborne. Improving 
requirements tracing via information retrieval. In Proc. 11th RE, pages 

[5] Jane Cleland-Huang, Orlena CZ Gotel, Jane Huffman Hayes, Patrick 
Mader, and Andrea Zisman. Software traceability: trends and future 


[7] Annibale Panichella, Bogdan Dit, Rocco Oliveto, Massimiliano 
Di Penta, Denys Poshvyvanyk, and Andrea De Lucia. How to effectively 
use topic models for software engineering tasks? an approach based on 


[9] Abdallah Qusef, Gabriele Bavota, Rocco Oliveto, Andrea De Lucia, 
and Dave Binkley. Recovering test-to-code traceability using slicing 
and textual analysis. Journal of Systems and Software, 88:147–168, 
2014.

[10] Marie-Catherine De Marneffe, Bill MacCartney, Christopher D 
Manning, et al. Generating typed dependency parses from phrase 

distance and applications. IEEE Transactions on Pattern Analysis and 

D. Poshvyvanyk, and A. De Lucia. When and how using structural 
information to improve ir-based traceability recovery. In Proc. 17th 

De Lucia, and Ettore Merlo. Recovering traceability links between 
code and documentation. IEEE Transactions on Software Engineering, 

[14] Xuchang Zou, Raffaella Settimi, and Jane Cleland-Huang. Phrasing in 
dynamic requirements trace retrieve. In Proc. 30th COMPSAC, pages 

[15] Jane Cleland-Huang, Brian Berenbach, Stephen Clark, Raffaella Settimi, 
and Eli Romanova. Best practices for automated traceability. Computer, 

[16] Jane Huffman Hayes, Alex Dekhtyar, and Senthil Karthikeyan 
Sundaram. Advancing candidate link generation for requirements 
tracing: The study of methods. IEEE Transactions on Software Engineering, 

[17] Jane Cleland-Huang, Raffaella Settimi, Oussama BenKhadra, Eugenia 
Berezhanskaya, and Selvia Christina. Goal-centric traceability for 
managing non-functional requirements. In Proc. 27th ICSE, pages 362– 
371, 2005.

Enhancing an artefact management system with traceability recovery 

[19] Alex Dekhtyar, Jane Huffman Hayes, Senthil Sundaram, A Holbrooke, 
and Olga Dekhtyar. Technique integration for requirements assessment. 

2014.

[21] Jun Lin, Chan Chou Lin, Jane Cleland-Huang, Oussama Ben Khadra, 
Ettore Merlo. Recovering traceability links between 
requirement-to-code and documentation. IEEE Transactions on Software Engineering, 

[22] Jane Cleland-Huang, Raffaella Settimi, Oussama Ben Khadra, Eugenia 
Berezhanskaya, and Selvia Christina. Goal-centric traceability for 
managing non-functional requirements. In Proc. 27th ICSE, pages 

Enhancing an artefact management system with traceability recovery 

[24] Alex Dekhtyar, Jane Huffman Hayes, Senthil Sundaram, A Holbrooke, 
and Olga Dekhtyar. Technique integration for requirements assessment. 

Software traceability with topic modeling. In Proc. 32nd ICSE, pages 

[26] Jun Lin, Chan Chou Lin, Jane Cleland-Huang, Raffaella Settimi, 
Joseph Amaya, Grace Bedford, Brian Berenbach, Oussama Ben Khadra, 
Chuan Duan, and Xuchang Zou. Poirot: A distributed tool supporting 
enterprise-wide automated traceability. In Proc. 14th RE, pages 

[27] Yonghee Shin and Jane Cleland-Huang. A comparative evaluation of 
two user feedback techniques for requirements trace retrieval. In Proc. 

[28] Wei-Keat Kong, Jane Huffman Hayes, Alex Dekhtyar, and Olga 
Dekhtyar. Process improvement for traceability: A study of human 

[29] David Cuddeback, Alex Dekhtyar, Jane Huffman Hayes, Jeff Holden, 
and Wei-Keat Kong. Towards overcoming human analyst fallibility in 
the requirements tracing process (nier track). In Proc. 33rd ICSE, pages 
Information for Authors

Scope

The aim of the International Journal of Computer & Information Science is to present information on well defined theoretical results and empirical studies, to describe current and prospective practices and techniques, to provide critical analysis of the use of CASE tools and to present broad issues in the fields of Computer Science and Engineering, Software Engineering and Information Science and Engineering. A primary aim of the journal will be to make the research accessible and useful to practitioners through carefully written methodology section and discussions of research results in ways that relates to practitioners.

Submission Policy

1. The IJCIS does not accept manuscripts or works that have been published in a journal, but accepts substantially revised and expanded versions of papers that were previously presented at conferences. In this case, the author should inform the Editor-in-Chief of the previous publication when the paper is submitted.

2. Submissions, referencing and all the correspondence will be conducted by e-mail.

3. Manuscripts should be in English and are subject to peer review before selection.

Procedure for Submission for Review

1. Send your complete text of manuscript to the Editor-in-Chief (e-mail: lee@cps.cmich.edu) via e-mail as an attachment document either in Word, PDF or PostScript format.

2. It may be useful for non-English speakers to have a native English speaker read through the article before submission.

3. Your e-mail message (to the Editor-in-Chief) should include the title of your paper, author(s) information (department/section, university/company, phone/fax numbers, e-mail address), the corresponding author’s complete address (mailing address), and area(s) of the paper.

Style of Manuscript to be Submitted for Review

1. The text should be double-spaced, two columned, single-sided in Times 12-point type on 8.5-inch x 11-inch paper with margins of 1 inch on all four sides. It should not exceed 30 double-spaced pages. All pages must be numbered sequentially.

2. Provide a 200-250 word abstract, 5-7 keywords. Your abstract should briefly summarize the essential contents.

3. All figures/tables must be captioned, numbered and referenced in the text.

4. An acknowledgement section may be presented after the conclusion section.

5. References should appear at the end of the paper with items referred to by numbers in square brackets.

Journal Papers: author’s first initials, followed by last name, title in quotation marks, journal (in Italics), volume and issue number, inclusive pages, month and year. For e.g.:


Papers of Proceedings: author’s first initials, followed by last name, title in quotation marks, proceedings / symposium / workshop name (in Italics), location, month, and year, and inclusive page numbers. For e.g.:


Books: author’s first initials, followed by last name, title (in Italics), edition, location: publisher, year. For e.g.:


Accepted Manuscript Submission

All accepted papers are edited electronically. You are required to supply an electronic version of the accepted paper. The detailed guidelines for preparing your final manuscript will be sent upon article acceptance. It is also available on the ACIS website at http://acisinternational.org.

Page Charges

After a manuscript has been accepted for publication, the author will be invoiced a page charge of $400.00 USD for up to 10 pages if he/she is not a member of ACIS and thereafter an additional $30 per printed page to cover part of publication after the receipt of your final manuscript (camera ready).