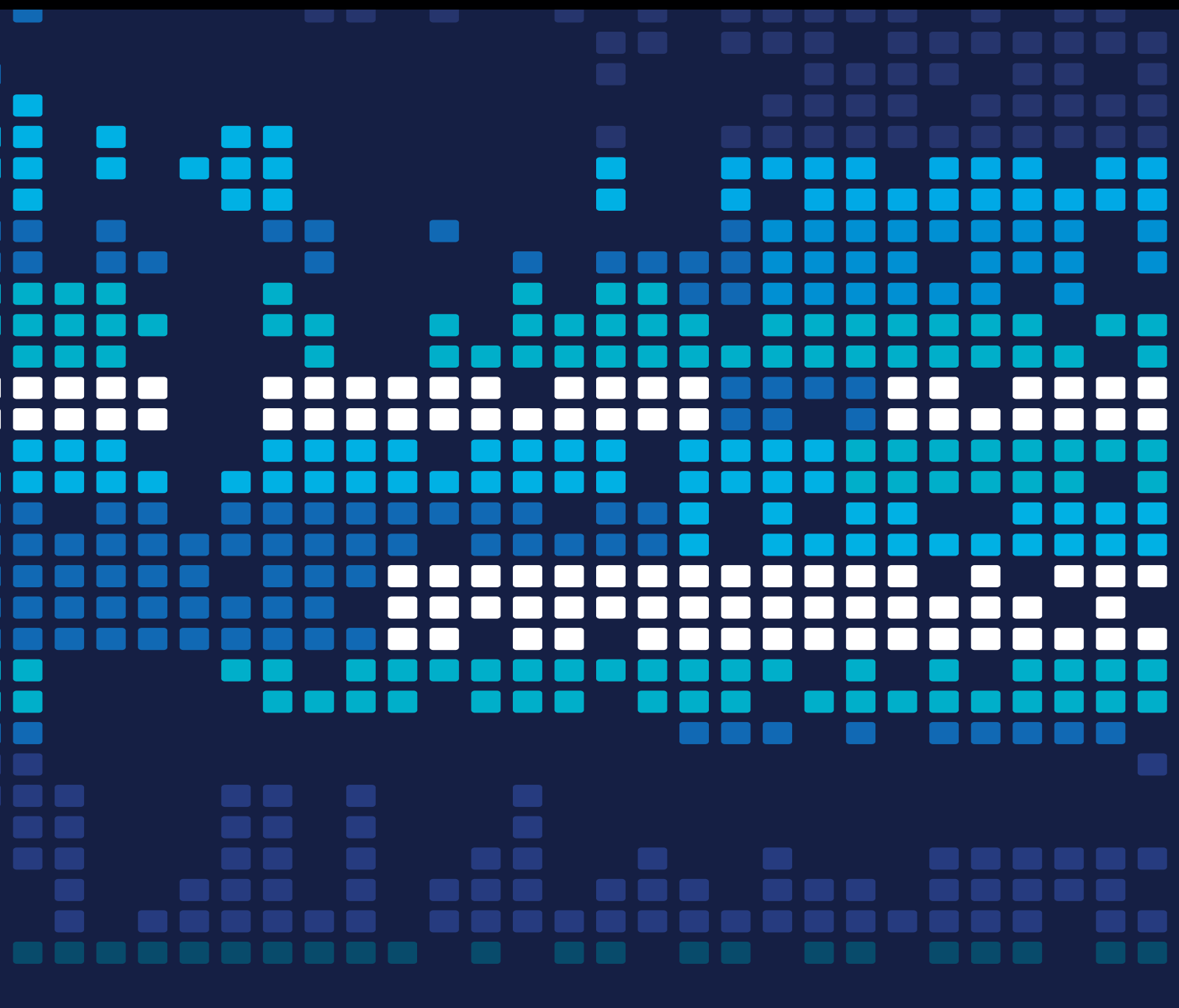
Scientific Programming

Advances in Big Data Programming and System Software

Lead Guest Editor: Raimund Kirner
Guest Editors: Xuanhua Shi, Xiaoming Li, and Alex M. Kuo



Advances in Big Data Programming and System Software

Scientific Programming

Advances in Big Data Programming and System Software

Lead Guest Editor: Raimund Kirner
Guest Editors: Xuanhua Shi, Xiaoming Li, and Alex M. Kuo

Editorial Board

Contents

Hindawi
Scientific Programming
Volume 2017, Article ID 1901876, 18 pages
https://doi.org/10.1155/2017/1901876

Research Article

Enhancing Health Risk Prediction with Deep Learning on Big Data and Revised Fusion Node Paradigm

Hongye Zhong and Jitian Xiao

School of Science, Edith Cowan University, Joondalup, WA, Australia

Correspondence should be addressed to Hongye Zhong; hzhong@our.ecu.edu.au

Received 5 January 2017; Revised 14 April 2017; Accepted 9 May 2017; Published 28 June 2017

Academic Editor: Michele Risi

With recent advances in health systems, the amount of health data is expanding rapidly in various formats. This data originates from many new sources including digital records, mobile devices, and wearable health devices. Big health data offers more opportunities for health data analysis and enhancement of health services via innovative approaches. The objective of this research is to develop a framework to enhance health prediction with the revised fusion node and deep learning paradigms. Fusion node is an information fusion model for constructing prediction systems. Deep learning involves the complex application of machine-learning algorithms, such as Bayesian fusions and neural network, for data extraction and logical inference. Deep learning, combined with information fusion paradigms, can be utilized to provide more comprehensive and reliable predictions from big health data. Based on the proposed framework, an experimental system is developed as an illustration for the framework implementation.

1. Introduction

The tendency of digital healthcare solutions is to transform the entire healthcare process in more flexible and efficient patterns. The most common applications of digital healthcare include electronic health records, mobile devices, and wearable health devices.

Electronic health records are initially sourced from health checks and diagnostic data of patients. Digitalization facilitates the sharing of the health records across different medical organizations. With these digitalized records, doctors can have a better understanding of the medical history of their patients [1]. However, when information accumulates over time, health records proliferate in large volumes. This causes difficulties for processing, storage, and retrieval. Estimates suggest that health records data may reach 12 ZBs by 2020 [2].

With the prevalence of smart phones, medical apps are available for many useful functions such as electronic prescribing, assessment, clinical decision support, treatment practice management, and self-care. Wearable health devices represent another rapid growth area which is transforming traditional healthcare to active and continuous health management. Research estimates that the number of wearable health devices could reach 169.5 million globally by 2017 [3]. The physiological sensors embedded in the wearable devices allow for capturing of additional health data. For example, heart rate and blood pressure can also be detected by smart phones.

With the growth of big data, scientists have found increasing value in deep learning and information fusion for the analysis of large volume of dynamic data. Deep learning applies a set of machine-learning algorithms at multiple levels to exploit the different layers of nonlinear information [4]. Information fusion applies continuous processes that collate relevant information to achieve situation awareness, which can support decision-making [5].

With the advent of global healthcare challenges such as age-related health problems and chronic diseases, researchers are actively seeking innovative solutions for disease prevention and health diagnoses in more efficient and economical ways. Using information fusion techniques with the digital health data to produce a viable solution has become a key topic of the industry. The research aims to provide an enhanced framework for developing health risk prediction systems by combining the techniques of deep learning and information fusion.

The rest of the paper identifies the research potential of deep learning and big health data (Section 2), reviews the preliminary methodologies related to the research (Section 3), and introduces an innovative framework to overcome some shortfalls in the peer research (Section 4). Finally, issues with experiment design and performance evaluation are discussed (Section 5).

2. Research Potential

2.1. Deep Learning on Big Health Data. Scientists have started to realize the insights of the health data held by various repositories and to make decisions for healthcare. However, most of the development potential is still in its infancy. The overall theories and techniques for predictively modeling and analyzing health data have not been adequately developed yet [6]. The conventional data mining methods suffer from accuracy and efficiency issues due to the constraints of data size and data quality [7]. Many conventional methods cannot directly fit in the big data context. In some environments, the concepts are so complex that they have to be learnt by sophisticated procedures. In particular, when there are underlying functional dependencies within a complex nature, we are unable to analytically express the data model in a simple way [8]. The industry is calling for advanced methodologies that allow information extraction from unstructured health data in large volumes [9].

Conventional data mining or machine-learning methods depend heavily on the representation of the data. In complex environments, it is difficult to extract high-level and abstract features for data representation due to variant factors [10]. Deep learning is introduced to solve this problem whereby it represents a complex concept by combining several simpler concepts. Deep learning is a machine-learning paradigm with a set of algorithms that attempt to learn a complex matter by mapping it to different levels of abstraction [4]. It has also been proposed for medical knowledge modeling and assisting in decision-making on healthcare issues. It is valuable for constructing health diagnosis models in a cost-efficient manner. In contrast, the cost of building an expert health system in the conventional approach was usually unaffordable [11].

Various computing models are being rapidly developed and applied in deep learning. Neural network is one of the most common models that can be used for implementing deep learning systems. They are highly adaptable to feature extraction from various situations [12].

Although deep learning can extract features from complex data, in a big data context, data combination must be performed before analytics take place. As information fusion can provide various methodologies for merging data from multiple sources, embracing the concept of information fusion within deep learning domain can enhance big data analytics with more systematic designs and more efficient processes.

2.2. Enhancement for AIA Vitality. AIA is an insurance company that owns a large customer base over Asian-Pacific countries including Australia and China. AIA recently introduced a science-backed wellness program called AIA Vitality that attempts to motivate the customers to maintain their health status with rewards and benefits provided by the partner organizations (see Figure 1).

The members of AIA Vitality can understand their current health status by uploading their health figures from their mobile and health devices, performing health assessments at the partner medical centers, or receiving periodic health reports generated from the collected formation. The members can improve their health by participating in health activities and meeting the health goals proposed by the system. Health reports will be generated periodically for members by analyzing the health records collected from different channels and the health activities undertaken by members. Based on their health status, members can be rewarded by the partner organizations.

The health risk predictions contained in the health reports mainly come from the expert opinions of the partner medical centers and in-database analysis rules. Health experts can log in to the system, review the health records of the members, and input their opinions. However, this process has been criticized for the overconsumption of both money and time. The health prediction rules set in the database are triggered when the health figures of a member meet certain conditions. The triggered rule will lead to a health risk being added in the report. For example, when the blood glucose of a member at a certain age reaches a certain level for a certain duration, the risk of diabetes will be identified in the member's health report. However, this process can only provide limited prediction categories with insufficient accuracy. We propose overcoming these issues and enhancing the health prediction by applying deep learning and information fusion on the big data in the system. Applying deep learning to big health data, the hierarchy of features can be learnt by establishing high-level features in terms of health prediction from low-level health datasets [13].

3. Preliminary Methodologies

To date, the healthcare industry has realized the potential benefits that can be gained from big data analytics. Recently, it has attracted extensive research on the architectural design and implementation strategies of big data analytics in the area of health data.

3.1. Architectural Design. The fundamental requirement of big data analytic systems is to deal with the volume, variety, and velocity of data coming from sources sharing the same context. Heterogeneous data is generated and stored in multiple sources such as relational databases, XML (Extensible Markup Language) documents, and RDF (Resource Description Framework) [3]. Transforming these heterogeneous datasets into understandable and sharable knowledge requires services that collect, prepare, and process these datasets.

The architectural design for a big data analytic system in healthcare is similar to that of a traditional big data analytics project. NIST (National Institute of Standards and Technology) provides a reference on architectural design for

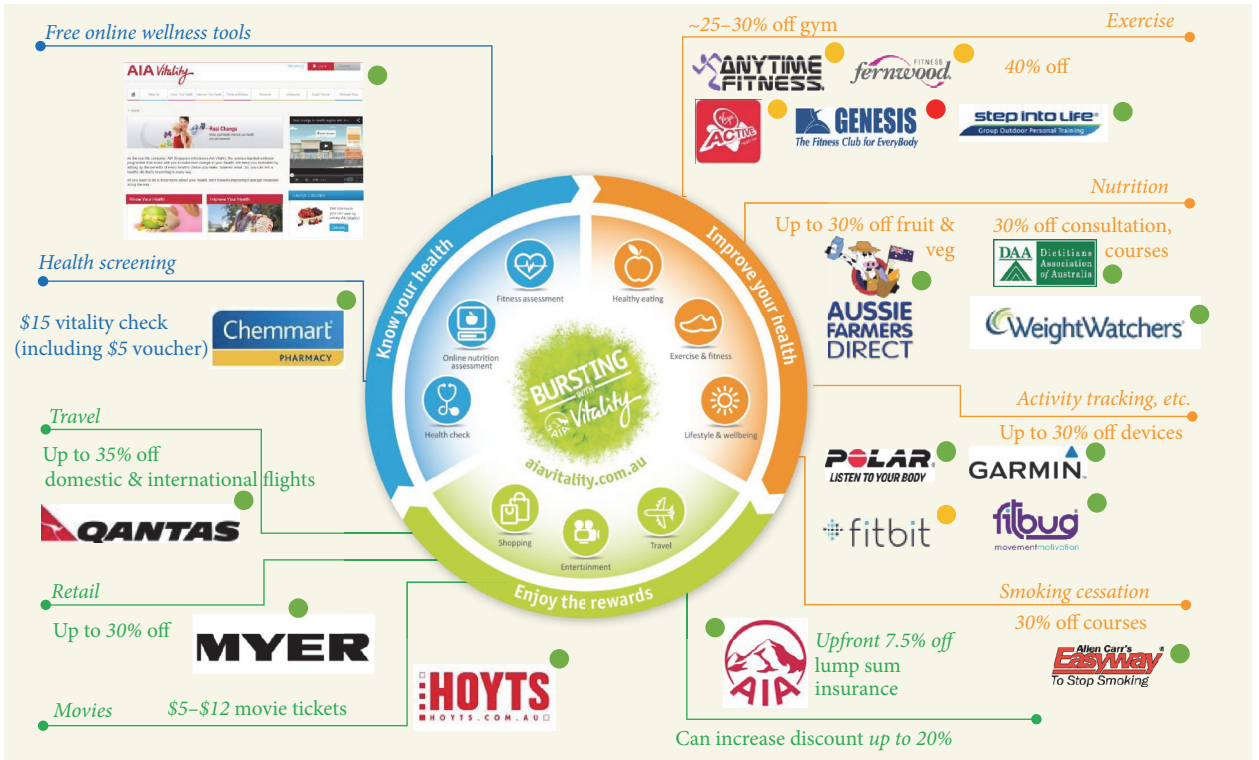


Figure 1: AIA Vitality program.

big data systems [14]. There have been several attempts to adapt the NIST reference to healthcare domains. Among those, one worth mentioning is the architectural design for big data healthcare analytics by Wang et al. (see Figure 2) [15]. This design functionally divides a big data analytic system into five layers, namely, (1) Data Layer, (2) Data Aggregation Layer, (3) Analytics Layer, (4) Information Exploration Layer, and (5) Big Data Governance Layer.

Data Layer contains all the data collected from various sources that can provide the insights to support daily operations and solve business problems. The data from various sources can be structured (e.g., database records) or unstructured (e.g., log files or images).

Data Aggregation Layer is responsible for data acquisition from the various data sources and transformation of the data into a specific standard format that is suitable for data analysis. Data transformation is often a major obstacle for implementing big data, as the data characteristics can vary dramatically.

Analytics Layer is responsible for performing appropriate analytics on the aggregated data based on the fundamental analysis goals. Recent advances in cloud computing and big data analysis, respectively, provide distributed data storage and computing techniques, including MapReduce, stream programming, and in-database analytics, which allow for the handling of vast amounts of data in a more reliable and efficient manner [3].

Information Exploration Layer generates outputs to make the analytic results comprehensible for users. The outputs can be visualization reports, monitoring of information, or health predictions to help users make decisions.

Big Data Governance Layer is a component that manages all the other logical layers. Its functions include data accessibility, data lifecycle, and data security managements. These functions ensure the availability of the data for processing and protect the privacy of the information owners.

3.2. Data Mining Methods. Data mining is the process of information extraction and pattern discovery from large amounts of data. This concept is expanded by machine learning to transform data into intelligent action [16]. Data mining is being used in the emerging healthcare area to manipulate clinical and diagnostic data and thereby provide reliable disease detection and healthcare systems [7]. The data mining methods used in the health industry fall predominantly into two categories: cluster analysis and classification. They provide mathematical foundations for the processes of learning and prediction.

Cluster analysis is an unsupervised machine-learning method that can organize a group of objects with similar characteristics into distinct categories. The main aim of clustering is to find structure within a given set of data [17]. Clustering is useful in healthcare for identifying the association between risk factors and health. For example, the research of Vermeulen-Smit et al. applied clustering patterns to explore the relationship between health risk behaviors and mental disorders [18]. Clustering is also useful for anomaly detection, revealing the most anomalous or unusual data from the cohort dataset.

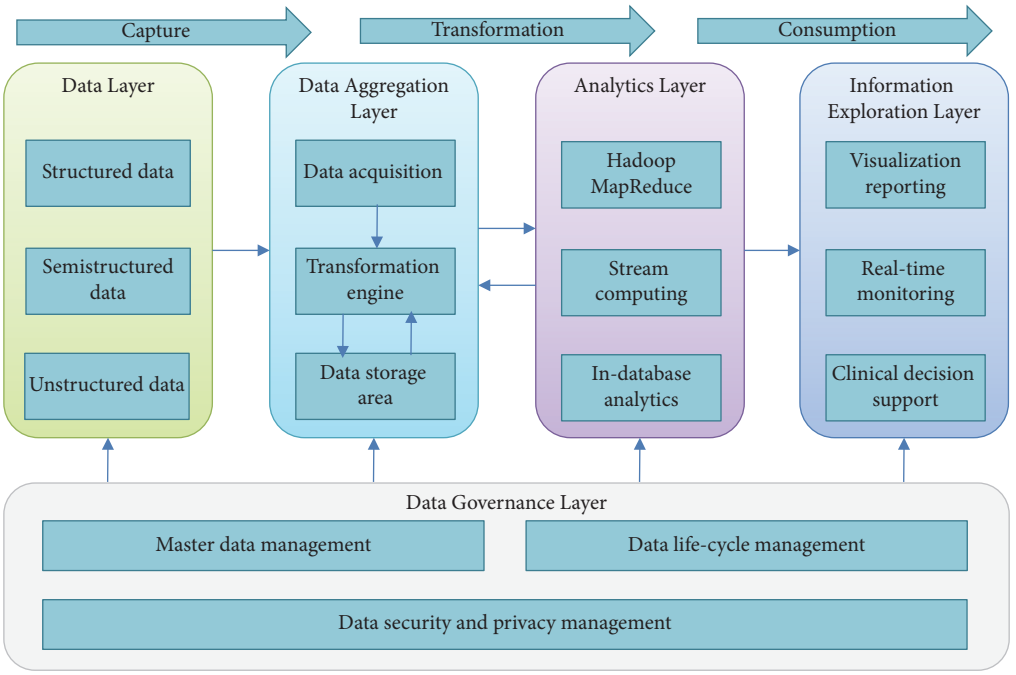


Figure 2: Big data architecture for healthcare.

Classification is a supervised learning process for prediction of the class for a given unlabeled item in a finite set of predefined classes based on a training set of data containing observations [19]. The most common classification algorithms include Decision Tree, K-Nearest Neighbor (KNN), Logistic Regression (LR), Support Vector Machine, and Bayesian Classifier. Classification is useful for predictive learning in health fields. For example, Bayesian Classifier was introduced by Kazmierska and Malicki to the diagnosis of brain tumor patients and help in the clinical treatment of the patients [20].

Decision Trees can classify data items into a finite number of predefined classes. K-Nearest Neighbor classification assigns labels of predefined classes to the test items by finding a group of K items in the training set in the neighborhood. Logistic Regression computes a linear combination for the input item by the logistic function with a given set of features. Support Vector Machine separates the test items by finding the separating lines with maximum distance to the points of the different groups of items [21].

Bayesian Classifier is a supervised and probabilistic learning method. The training data is used to calculate an observed probability of each outcome based on the evidence provided by feature values. The observed probabilities are then applied to predict the most likely class for the new features [16].

3.3. Bayesian Fusion and Neural Network. By consolidating between different information sources, information fusion can perform data mining to reduce uncertainty and achieve a better understanding of the information. By adopting Bayesian statistics, it can provide several methods for information fusion [22], for instance, the Bayesian Classifier above.

Bayesian fusion algorithms use the knowledge of Bayesian Inference to make inferences about the identity of events in the observation space [23]. Bayesian Inference is the process that applies a probability model to a set of data and evaluates the resulting probability distribution for the fit of the model towards the observed and unobserved data [24]. The Bayesian Inference process can be applied to the fusion from multiple information sources.

In the inference process, a.k.a. Bayesian fusion process (Figure 3), each of the data sources provides a hypothesis on the observation and the source-specific algorithm. The hypothesis $H^n(k = 1, 2, \ldots, n)$ is used to estimate the probability of the type of each entity by the likelihood function $P(H^k \mid O_i)$, where i denotes the fact type and O_i the entity from a data source. The probabilities of the data sources will be combined by using the Bayesian Inference Function. The output of the inference is the combined probability $P(H^1, H^2, \ldots, H^n \mid O_i)$. Decision Logic is used to optimize the combined probability when there are constraints (e.g., thresholds) on the final output. The final output of the process is called the Fused Identity of O_i.

The ground of the Bayesian Inference is Bayes' Rule, which is a probability-based reasoning discipline. Bayes' Rule can be derived by evaluating the probability of hypothesis H conditioned on knowing event E, which is defined as

$$P(H \mid E) = \frac{P(H, E)}{P(E)}, \tag{1}$$

where $P(H, E)$ denotes the probability of the intersection of hypothesis H and event E [23].

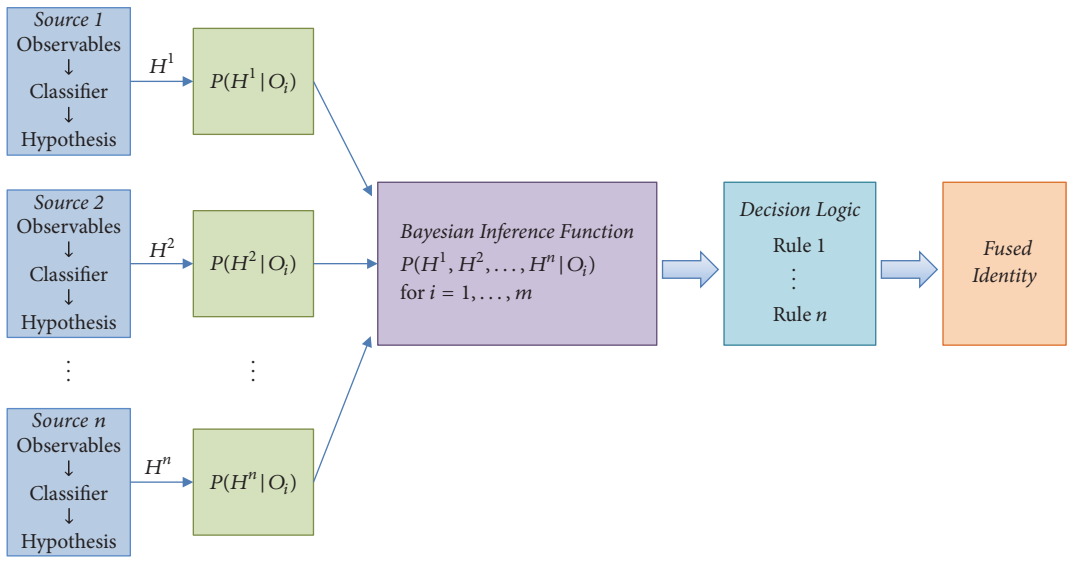


Figure 3: Bayesian fusion process.

The inference applied in Bayesian Classifier can be defined as

$$\text{Assign } x \text{ to } C_i = \arg\max_{C_i} P(C_i \mid x), \tag{2}$$

where C_i $(i = 1, 2, \ldots, m)$ denotes a given set of m classes, $P(C_i \mid x)$ denotes the respective posterior probabilities, and x represents an unknown feature vector. The unknown attribute x is identified by assigning it to the class for which the posterior probability becomes maximum [8].

Bayes' Rule enables the correct estimation of a prior probability for an unknown event when the probability of the evidence about the event can be found. This works on the assumption that the unknown event and the evidence are independent. However, in the real world there are many unknown events and different types of evidence, some of which are related to each other [25]. Therefore, the Bayesian Network is introduced to represent the joint probability model among given variables by a directed acyclic graph, where the nodes of the graph represent random variables and the directed edges represent direct dependencies between the variables [26]. Mathematically, this type of structure is called a directed acyclic graph (DAG). The join probability of the model can be calculated by

$$\begin{aligned} P(U) &= P(X_1, \ldots, X_n) = \prod_{X_i \in U} P(X_i \mid X_1, \ldots, X_n) \\ &= \prod_{i=1}^{n} P(X_i \mid \text{Pa}(X_i)), \end{aligned} \tag{3}$$

where $U = \{X_1, \ldots, X_n\}$ is the set of random variables on interest called universe and $\text{Pa}(X_i)$ denotes the parental variables of variable X_i. Referring to a DAG, it represents the ith node in the graph corresponding to the factor $P(X_i \mid \text{Pa}(X_i))$ [27]. Bayesian Network is a fusion approach that is suitable for the uncertainty measurement for the extracted properties with graphical structure and probability calculus [28].

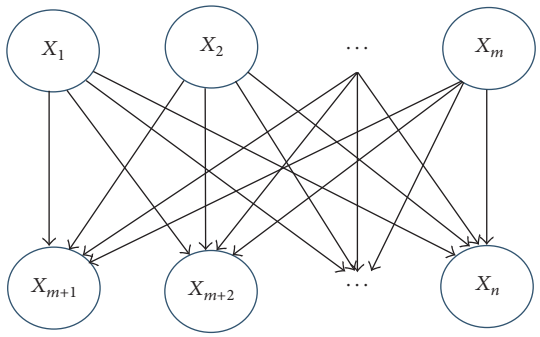


Figure 4: Example of Bayesian Network.

A Bayesian Network as shown in Figure 4 represents the association $\{X_1, X_2, \ldots, X_m\} \Rightarrow \{X_{m+1}, X_{m+2}, \ldots, X_n\}$, where the nodes X_i represent the variables and the edges represent the direct influence. The joint probability distribution of the network can be defined as

$$\begin{aligned} &P(X_1, X_2, \ldots, X_m, X_{m+1}, \ldots, X_n) \\ &= \prod_{i=1}^{n} P(X_i \mid \text{parents}(X_i)) \\ &= \prod_{i=1}^{n} P(X_i) \prod_{j=m+1}^{n} P(X_j \mid X_1, X_2, \ldots, X_m) \\ &= \prod_{i=1}^{n} S(\{X_i\}) \prod_{j=m+1}^{n} \frac{P(\{X_j, X_1, X_2, \ldots, X_m\})}{P(\{X_1, X_2, \ldots, X_m\})} \\ &= \prod_{i=1}^{n} S(\{X_i\}) \prod_{j=m+1}^{n} \frac{S(\{X_j, X_1, X_2, \ldots, X_m\})}{S(\{X_1, X_2, \ldots, X_m\})}, \end{aligned} \tag{4}$$

where S is the support of an itemset.

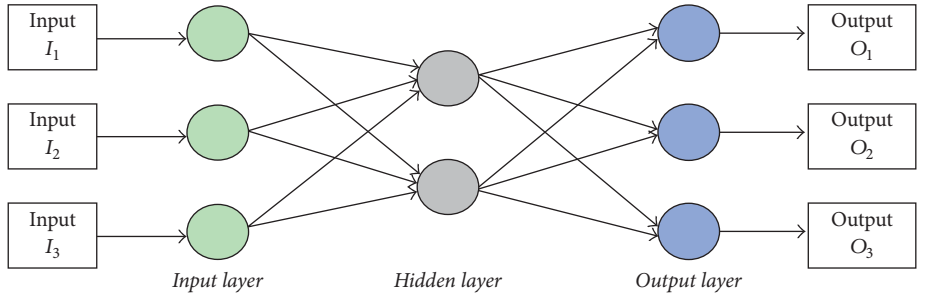


Figure 5: Three-layer feedforward network.

In information fusion, the selection of features can be very complex under certain circumstances. For example, a speech recognition task in a noisy environment is difficult due to the interference of irrelevant features [29]. On such occasions, feature extraction approaches can be used to extract the main features or characteristics of the situation [12]. Neural network can provide a flexible method to capture essential features from data containing noise and partial information [30].

Neural network is a function-approximation technique that with a set of given inputs and a desired set of outputs can be trained to map observed data to a function [31]. A standard structure for a neural network is a multilayer feedforward network [12].

A simple example is a three-layer perceptron network (Figure 5), which consists of the input layer, the hidden layer, and the output layer. The nodes in the input layer supply the input signals to the nodes in the hidden layer. The connections between the nodes in the input layer and the hidden layer represent the weights between the two layers, which need to be determined by training. The output signals of the hidden layer are used as inputs to the third layer. The outputs of the nodes in the output layer constitute the network output. The network output is usually an output prediction, a function approximation, or a set of classification outputs [30].

Each node in the hidden layer simulates a brain neuron, which is composed of a summation function and an activation function (see Figure 6). The summation function computes a linear combination of the weighted input signals [27]. The function can be defined as

$$u_j = \sum_j (\cdot) = \sum_{i=1}^{n} w_{ij} v_i + b_j, \tag{5}$$

where v_i denotes the ith input signal in the total n input signals, w_{ij} denotes the weight of the input signals associated with the jth hidden node, and b_j is the externally applied bias.

The output of the summation function is passed to the activation function that squashes the permissible range of the output signal to a finite value. The activation function is introduced to provide nonlinearity into the network, which is usually differentiable and chosen depending upon the requirements of the case [32].

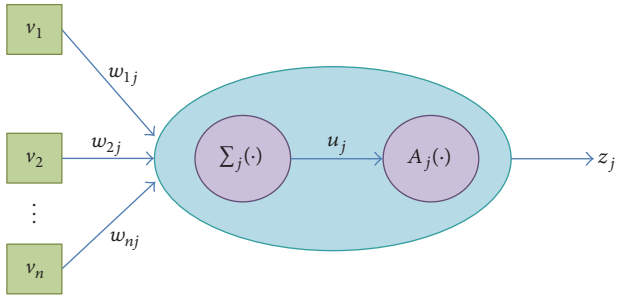


Figure 6: Neural node structure.

4. Deep Learning Fusion Framework

Deep learning is not an entirely new technology. Examples of the application of deep learning in healthcare can be found in several publications, for instance, on mammogram analysis [33] and video pornography detection [34]. However, these applications are mostly limited to image analysis. Additionally, these studies have not been widely adopted, as the architecture of deep learning is more complex than conventional data mining systems, and the previous studies failed to propose system designs that are compatible with the analytics in a production environment.

Deep learning systems require more computing power and memory storage. Recently, with the broad uptake of cloud computing, deep learning has been given new impetus by offloading the computation onto the cloud [35]. This allows an analytic system with complex learning algorithms to be hosted on the cloud in a cost-efficient manner.

A multilayered architecture is used in deep learning to map the relations between inputs features and the outcomes. This architecture makes deep learning more suitable for analyzing a large number of variables. The model containing many levels of nonlinearities excels at identifying complex patterns in data [32]. However, in a big data environment, data that comes from multiple sources and from multiple structures need to be fitted into multiple models. The fusion of the models should be performed at a specific level of the architecture to achieve the overall learning purpose.

The most fundamental fusion model is called situation awareness (SAW). SAW can be divided into three layers, in terms of (1) perception of elements in current situation,

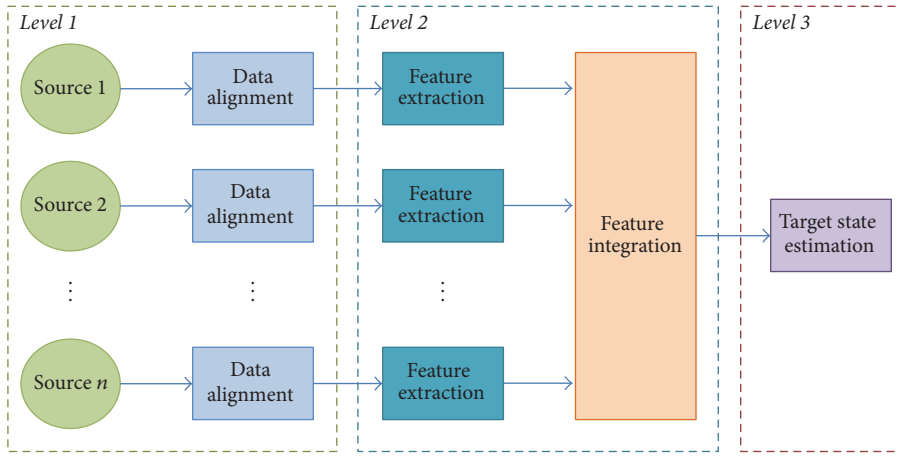


Figure 7: Deep Health Analytic Framework.

(2) comprehension of current situation, and (3) projection of future states [36]. The research of Sundareshan and Wong [37] extended the concept to the extraction of features from multiple data sources. The research of Mitchell [38] developed a parallel network for the fusion of multiple data inputs. These research projects provide useful guide on structuring data analysis system. However, they lack details on implementing the data process components. Steinberg et al. introduced the fusion node paradigm that indicates the main processes contained in a fusion system in terms of fusion, association, and alignment [39]. This paradigm has proven to be useful for developing automatic fusion systems that involve selecting the data flow among the fusion nodes (i.e., how to batch data for association and fusion processing). We integrate deep learning with the fusion processes to perceive the current situation and predict the future status of an individual's health from the big data.

Integrating fusion paradigms with preliminary big data techniques, we propose a framework suitable for deep learning on big health data, namely, the Deep Health Analytic Framework (DHAF, Figure 7). The framework is composed of three levels, which correspond functionally to the 3-layer structure of the SAW model. At the first level, the fusion processes aim to acquire the status, attributes, and dynamics of the relevant information elements in the environment. The processes consist of the data acquisition from multiple data sources and the alignment of the raw data. The eHealth data usually consists of various file objects, including images and PDF files. Although there are several commercial ETL (Extract, Transform, and Load) software programs, for example, Informatica, which allows the integration of data from multiple sources, the existing ETL systems are usually very costly and provide limited support of data types. The data alignment using machine-learning approaches is introduced to provide a more flexible solution. The proposed alignment processes involve classifying the information elements and restructuring them to representations compatible for further analysis [12]. The data alignment for each source is achieved by a fusion node that encompasses several components as described in Figure 8.

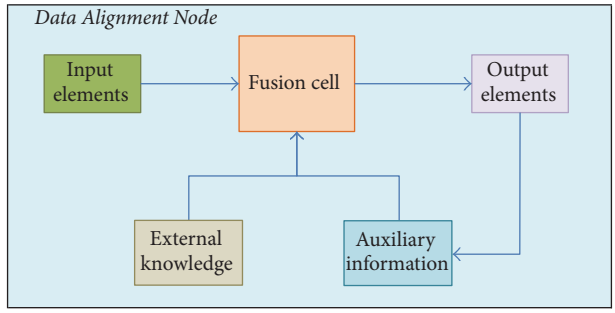


Figure 8: The fusion node of data alignment.

The input elements are the original information objects from a data source. Fusion cell is the smallest granular component of information fusion systems. It acts as an intermediate conceptual level between the different individual information elements from the data source and the global information processing systems [40]. Fusion cell refines the input data elements with the constraints or indicators set in the external knowledge and auxiliary information and ensures the quality of the output elements meet the requirements for further fusion processing. The external knowledge contains operation rules that can constrain and guide the fusion process. These rules contain the information required to process data in specific formats. For example, if the input data is in XML format, fusion cell will use the XML specification information for the data analysis. The auxiliary information stores additional contextual information to support the fusion process. The results derived from the fusion processes can also be recorded in auxiliary information to enhance future fusion.

The process method of fusion cell is usually a classifier, and the outputs are maps of classes and of confidence levels [41]. We use neural network to construct the fusion cell. According to the research of Amirani et al., we can compute Byte Frequency Distribution (BFD) and Principal Component Analysis (PCA) to identify different data types. BFD is the occurrence pattern of each byte value within

a file content. It can be used to discover the files derived from different data sources by detecting the file type fingerprints. PCA is a multivariate analysis technique that describes data with an orthogonal transformation of the coordinates. It can represent an approximation of a data structure with a subset of its primary components by dimensionality reduction [42]. A d-dimensional dataset $M = \{M_n \in R^d \mid n = 1, \ldots, N\}$ can be approximately represented by a smaller k-dimensional dataset $Z = \{Z_n \in R^k \mid n = 1, \ldots, k\}$. Z can be obtained by ME_k, which means projecting M onto the eigenvectors corresponding to the largest eigenvalues. E_k is the first k column of the eigenvectors E.

An autoassociative neural network is used for classifying the data type. A feedforward network is trained to produce an approximation of the identity mapping between inputs and outputs using back-propagation algorithm. The network has d input and d output nodes and a single hidden layer with k nodes and linear activations. In the training phase, the PCA projection matrix is calculated after the BFD of the input dataset is normalized. The normalization is to scale the data into the interval of $[-1, 1]$. The output dataset from PCA is fed to the neural network. As it is a nonlinear process, the hyperbolic tangent is chosen as the activation function. The function can be defined as

$$z = A(u) = 1 - \frac{2}{1 + \exp(2u)} \in (-1, 1), \quad (6)$$

where u denotes the input dataset and z the output dataset. The training outputs are compared with the desired outputs of the data types, and the synapse weights are adjusted accordingly to reduce the errors. After the training, the weight of the synapses is saved in the auxiliary information of the fusion node. Once the data type has been determined, the format specification in external knowledge will be used to verify the identification results, extract the metadata of the file object, and normalize the content of the object to a common representational format. If the identification is incorrect, an error will be recorded in auxiliary information to enhance future identification. For instance, an object that has been identified as a PNG image will be transformed to a standard size and color, and its content data and metadata will be stored in XML format. The metadata should contain information related to the environment such as the transaction time and spatial references. The transformation of the source data facilitates the analysis of information elements in a consistent set of units and coordinates for further processing [12]. The aligned data will be ready for the second-level fusion.

At the second level of the framework, fusion processes are intended to interpret and integrate the information elements under the current situation. The processes in this level include feature extraction and feature integration. The feature extraction is to acquire perceptual features containing the maximal information concerning the prediction objectives in the next fusion level. The extracted information is organized on the basis of the time sequence. Through the extraction process, the data quantity and data dimension are reduced to increase the efficiency and accuracy of further analysis. A fusion node for the feature extraction encompasses several components, as shown in Figure 9.

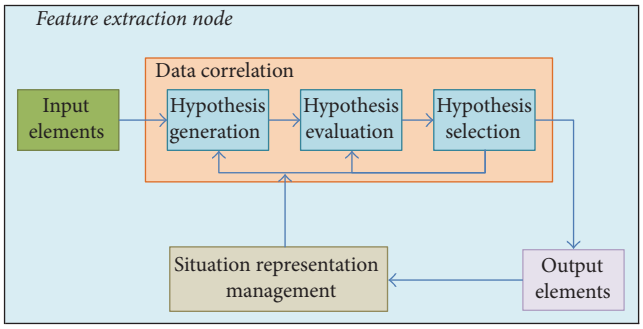


Figure 9: The fusion node of feature extraction.

The main process in the extraction node is Data Correlation. This determines which input elements associate with which elements currently in the situation representation being maintained by the node. The input elements for the association are the aligned data in the common format and the time coordinates that allow the elements to be tracked along with time. The correlation process is accomplished through three functions, namely, Hypothesis Generation, Hypothesis Evaluation, and Hypothesis Selection. Through the correlation process, the input data becomes normalized and fits the situation representation.

Hypothesis Generation identifies feasible association candidates that can be derived from the input elements [39]. In this function, the hypothesis of the associations between the input elements and the health risks will be generated. For example, running records taken from a wearable device can be used as the input elements, and a hypothesis can be generated to represent an association between the pattern of steps run and a health risk (e.g., obesity). Hypothesis Evaluation computes the confidence of the identified candidates based on the defined metrics to determine the associations. For example, the confidence of association between the running record and the obesity will be calculated.

The confidence calculation can be achieved by Bayesian Association Rule Mining algorithm [43]. For an equivalent timeframe, let A be an itemset (e.g., a health device record) consisted of items $I_1, I_2, \ldots, I_m$, and let R be a health risk consisted of items $I_{m+1}, I_{m+2}, \ldots, I_n$ (e.g., the historical diagnoses related to the health risk). The association $A \Rightarrow R$ can be represented in a Bayesian Network, where $I_1, I_2, \ldots, I_m$ are the Boolean variables in correspondence with $X_1, X_2, \ldots, X_m$ (see Figure 4) and $I_{m+1}, I_{m+2}, \ldots, I_n$ correspond to $X_{m+1}, X_{m+2}, \ldots, X_n$.

The confidence of association $A \Rightarrow R$ is defined as $P(R \mid A)$. The confidence can be computed with Bayesian Network, namely, Bayesian Confidence (BC), which can be represented as

$$\begin{aligned} \mathrm{BC}(A \Longrightarrow R) = P(R \mid A) &= \frac{P(A, R)}{P(A)} \\ &= \frac{P(X_1, X_2, \ldots, X_m, X_{m+1}, \ldots, X_n)}{P(\{X_1, X_2, \ldots, X_m\})} \\ &= \frac{P(X_1, X_2, \ldots, X_m, X_{m+1}, \ldots, X_n)}{S(\{X_1, X_2, \ldots, X_m\})}. \end{aligned} \quad (7)$$

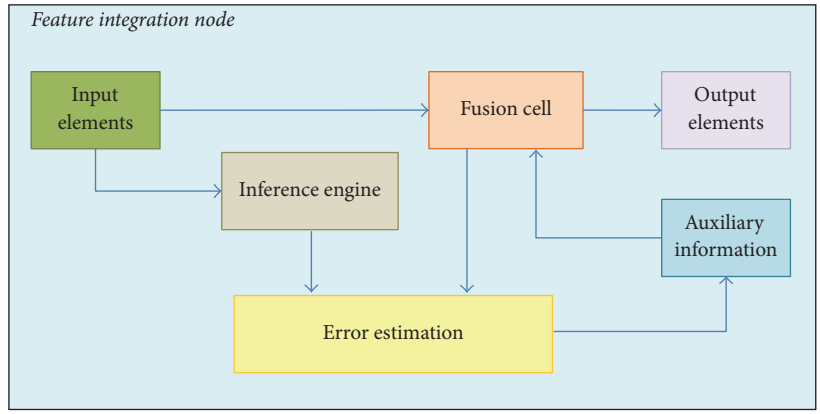


Figure 10: The fusion node of feature integration.

Hypothesis Selection determines which output associations the fusion system should retain and use for state estimation. For example, if the association of a hypothesis exists, that is, the calculated confidence value is greater than 1.0, the association should be updated into the situation model. The maintenance of the situation model is required to compute the probabilities of all the events, that is, the probabilities for the occurrence of various health crises. Let us denote the probability of a health risk as $P(R)$ and the conditions on the occurrence of a set of associations as $\{D_i \mid i = 1, \ldots, n\}$. If the itemset A introduced above is a selected association, we have $\{A \in D_i \mid i = 1, \ldots, n\}$. The probabilities of the events can be approximated to an aggregated probability $P(R \mid D_1, \ldots, D_n)$. The aggregation of the associations requires the computation of a probabilistic model of the joint distribution of $(R, D_1, \ldots, D_n)$, which can be achieved by building an approximation of the true conditional probability by the use of an aggregation operator. Based on Nu model [44], the approximation can be defined as

$$\begin{aligned} P(R \mid D_1, \ldots, D_n) &= \frac{P(R, D_1, \ldots, D_n)}{P(D_1, \ldots, D_n)} \\ &= \frac{P(R)\, P(D_1 \mid R)\, P(D_2 \mid R, D_1) \cdots P(D_n \mid R, \ldots, D_n)}{P(D_1, \ldots, D_n)} \\ &= P(R)\, \frac{\prod_{i=1}^{n} P\left(D_i, R, \overline{D}_{i-1}\right)}{P(D_1, \ldots, D_n)}, \end{aligned} \tag{8}$$

where $\overline{D}_{i-1} = \{D_1, \ldots, D_{i-1}\}$ is the set of all data up to the $(i-1)$th association. This model is useful for combining probability distributions conditioned to each individual association into a joint conditional distribution that can represent all original information. Besides Nu model, there are several alternative approximation methods, such as Tau model [44]. With this aggregation operation, we can compute the approximated probabilities for all the events in different time frames, which can represent the probabilities for various health risks in the eHealth system.

Each feature extraction node manages a fused situation representation for the data from its corresponding data source. The local fused situation representations for different data sources are to be integrated into a global situation representation, so that the developed model can be used for estimation and decision-making in the next fusion level. This process, namely, feature integration, is achieved by a deep learning method as described in Figure 10.

The input elements for the feature integration embrace the source data normalized by the nodes of data alignment and the probabilities of health incidents fused by the feature extraction nodes. Within a time frame where there are both inputs of normalized data and probabilities, the input probabilities from different sources will be combined to aggregated probabilities by the inference engine, and the input normalized data will be fed into the fusion cell for training purposes. Inference engine is a component that updates probabilities using inference scheme and evidence. Each piece of evidence updates the probability of a set of hypotheses calculated via Bayesian rule. The calculation can be defined as

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)}, \tag{9}$$

where A and B are events that are not necessarily mutually exclusive, $P(A \mid B)$ is the conditional probability of event A occurring given that event B has occurred [36]. If there are m mutually exclusive and exhaustive hypotheses $H_1, \ldots, H_m$ and n possible occurring events $X_1, \ldots, X_n$, the probability can be updated as

$$P\left(H_i \mid X_j\right) = \frac{P\left(X_j \mid H_i\right) P(H_i)}{\sum_{k=1}^{m} P\left(X_j \mid H_k\right) P\left(H_k\right)}. \tag{10}$$

Fusion cell in this node is a component where the input source data are classified with a convolutional neural network (CNN). CNN is a deep learning method more often applied in image recognition. The aim of the process of CNN is to map the n input aligned data to m output health risks in a vast sampling context with high performance (Figure 11). As introduced above, during the learning phase the construction of the approximation function of the neural network is mainly achieved by iteratively adjusting the values of the connection

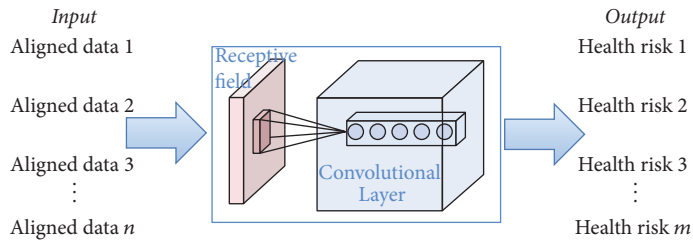


Figure 11: CNN-based health risk mapping.

weights of the nodes between different layers. With the optimized weight values, the approximation function is expected to produce outputs with fewer estimation errors. The training error can be calculated by comparing the differences between the outputs of the inference engine and the fusion cell. The calculation of the error can be defined as

$$E = \frac{1}{2N}\sum_{i=1}^{N}\left(t_i - z_i\right)^2, \tag{11}$$

where E denotes the error, t_i the target outputs from inference engine, z_i the predicted outputs from the neural network, and N the total number of training patterns [30].

The construction of the CNN is similar to the conventional neural networks introduced in the previous section, which are composed of neurons that have learnable weights and biases. Each neuron receives inputs and creates a dot product. The whole network expresses a single differentiate function that scores the input aligned data of health attributes in accordance with the classes of health risks. CNN has been proven to be more efficient in training inputs with a restricted number of parameters and hidden units. CNN can achieve local connections and tied weights efficiently by pooling translation invariant features. This specialty is adaptive to our design, as the input health data has been normalized and the output health risks have been predefined in certain classes [10].

Convolutional layer is the core building block of the network that performs most computations. The parameters of the layer consist of a set of learnable filters. Each filter has a small receptive field. In our context, this can be the subset of the input health documents, by putting the width, height, and depth of the imagery data into a 2D matrix. For acoustic data, we can cast the frequency and utterance of audio to receptive fields [45]. During the forward pass, each filter is convolved across the matrix of the input. The dot product is computed between the filter and the input, and a 2D activation map that represents the responses of the filter at every spatial position is produced. The network learns the activation of the filters when it detects some specific patterns at some spatial position in the input. Along time frames, the activation maps for all filters stack and form the output of the convolution layer.

Pooling layer is commonly inserted between successive convolutional layers in the architecture to progressively reduce the spatial size of the feature maps while retaining essential information retained. It also controls the overfitting of the neural network. The most common form of pooling

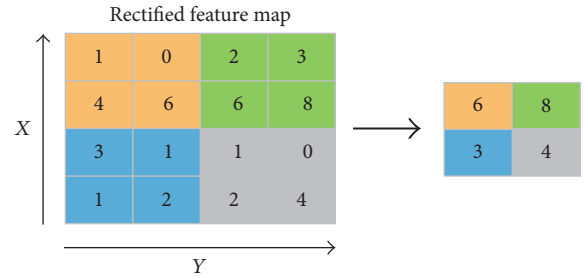


Figure 12: Max pooling example.

applies 2 × 2 filters with a stride of 2 at every slice in the input (see Figure 12). The largest elements within each filter are taken from the rectified feature map [46].

After several convolutional and max pooling layers, the high-level reasoning in the neural network is achieved via fully connected layer. Akin to the traditional multilayer perceptron (MLP), the neurons in the fully connected layer have complete connections with all activations in the previous layer. As a result, the activations can be computed with a matrix multiplication followed by a bias offset. The output of the connected layer is the probability distribution over labels, which can be defined as

$$P\left(y \mid U\right) = \frac{\exp\left(U \cdot w_j\right)}{\sum_{j=1}^{k}\left(U \cdot w_j\right)}, \tag{12}$$

where U denotes features produced by the pooling operation $U = \{c_1, c_2, \ldots, c_p\}$ (p is the number of features), k is the number of filters, and w denotes the weight matrix of the fully connected layer [47].

Putting all these together, we have the architecture illustrated as Figure 13. An additional operation called Introducing Nonlinearity, usually achieved via Rectified Linear Unit (ReLU), can be used after every convolution operation, which increases the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer. It computes the function $f(x) = \max(0, x)$. In other words, the activation simply has a threshold at zero.

At the third level of the fusion framework, at the time frame where no input probabilities can be provided, the neural network trained in feature integration can produce outputs for state estimation. In a timeframe where health records cannot be provided, a health estimation can be made

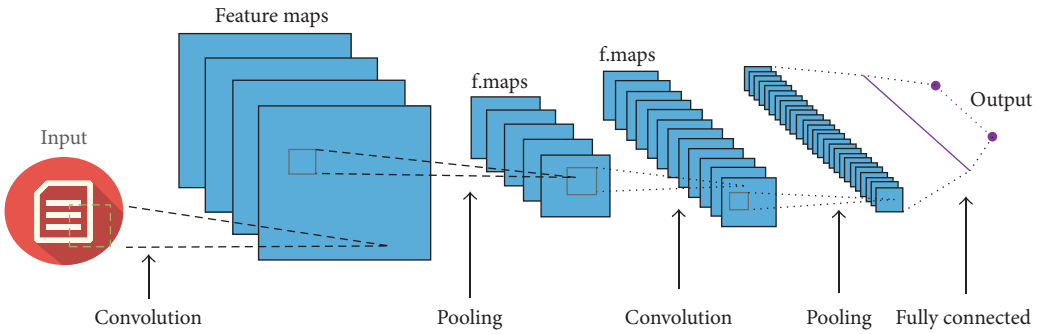


Figure 13: Overall CNN architecture.

based on the given inputs of normalized data. Note that the neural network cannot make predictions about future state where no input source data can be obtained. It can only make estimations for the missing previous or current states. With the estimated past and current probabilities of various health risks, time series analyses can be applied to predict the likelihood of future health risks. This process is called target state estimation in the framework. We propose to use ARMA (Autoregressive Moving Average) model for the prediction [48]. The model ARMA(p, q) can be defined as

$$X_t = \sum_{i=1}^{p} a_i X_{t-i} + \sum_{i=1}^{q} b_i w_{t-i} + w_t, \tag{13}$$

where a_i and b_i are the weights applied to the model, p is the autoregressive order, q is the moving-average order, and w_t is the white noise. ARMA(2, 1) is taken frequently to render a weekly stationary process for time series analysis, and the equation can be simplified as

$$X_t = a_1 X_{t-i} + a_2 X_{t-2} + b_1 w_{t-i} + w_t. \tag{14}$$

Fitting it with the past and current probabilities of the health risks, we are able to calculate the weights and white noise of the model. Thereby, the model can be used to predict the probabilities of future health risks.

5. Architecture Design and System Implementation

In the above section, we presented a conceptual framework that contains the essential processes for deep learning on eHealth data. This framework provides guidance on building an enhanced health analytic system. The proposed framework forms the basis for an architecture design which we have named Deep Health Analytic Architecture (DHAA). The architecture design is shown in Figure 14. In contrast to the architecture introduced by Wang et al. [6], DHAA applies the concepts of deep learning and information fusion to provide more reliable health prediction. Moreover, the new framework encourages MapReduce and SOA (Service-Oriented Architecture) paradigms to facilitate the distributed computing and systematic communication in enterprise environments.

In correspondence with DHAF, the architecture maps the three-level information fusion processes into four primary components, namely, data alignment, feature extraction, feature integration, and target state estimation. Data alignment consists of fusion nodes that normalize the raw data from different sources. Feature extraction consists of fusion nodes that fuse the normalized data for each data source. Feature integration updates the global situation representation and the estimation model of the neural network with three subcomponents in terms of inference engine, neural network Fusion, and error estimation. Target statement estimation performs health risk prediction based on the neural network model and regression prediction and generates analysis reports. The aggregation of the above four data analytic components are named as Analytic Engine. As an additional component in the architecture, Data Governance provides several functionalities including an input interface for raw data, cache for normalized data, storage for fusion knowledge bases, and access for the analysis reports. Data Governance is the sole interface for all the data to enter and exit the Analytic Engine. This centralized data management eases the utilization of distributed data storage and integrated data access through service calls.

In addition to the architecture, various techniques are applied for system implementation. The operating systems, programming languages, and software frameworks are carefully chosen to optimize the performance of the analysis. As a big data analytic system, the MapReduce paradigm is an important technique useful for enhancing analysis efficiency. MapReduce is a framework with the capability to process large datasets with a parallel, distributed algorithm in a cluster. MapReduce helps to overcome concurrency, robustness, scale, and other common challenges in the analysis for large volume of data [49], as it avoids most low-level implementation problems in distributed computing. As the name suggests, MapReduce mainly consists of a Map step and a Reduce step. Both the Map and Reduce steps can be distributed in a way that takes advantage of the multiple processor cores that are found in modern hardware to increase data process efficiency [50]. In the Map step, the data processing script is executed to produce new key/value pairs. The input data will be grouped on the key, and the value is the information pertinent to the analysis in the Reduce step. In the Reduce step, a function is executed once per key

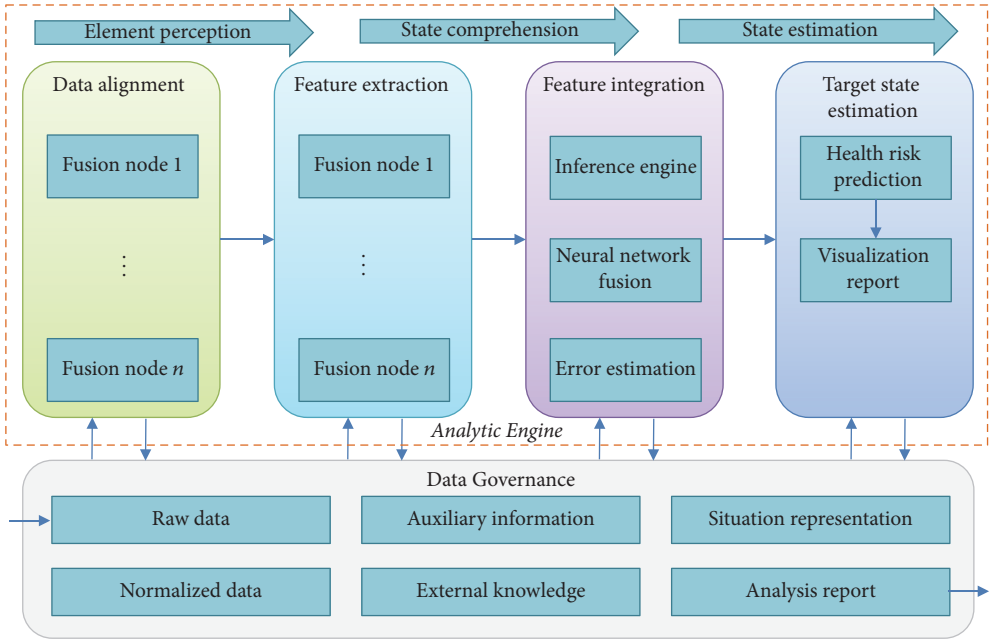


Figure 14: Deep Health Analytic Architecture.

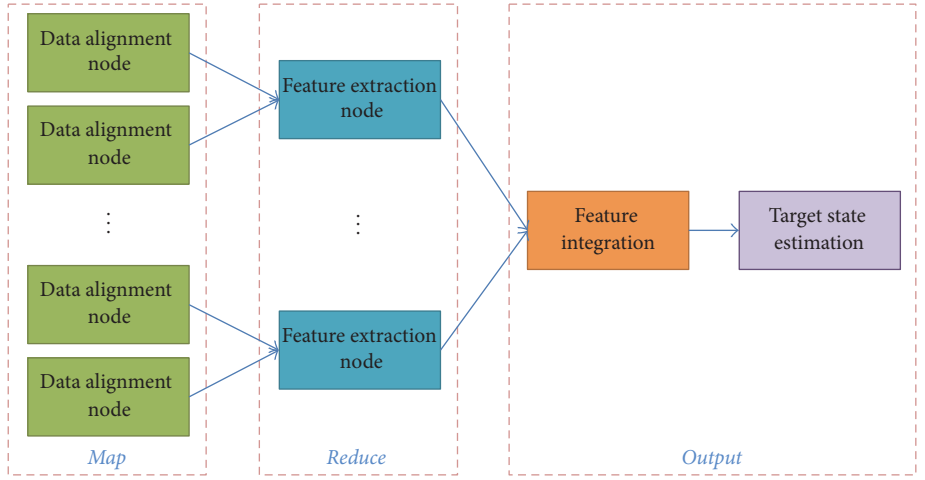


Figure 15: DHAA MapReduce pattern.

grouping, and it iterates over all of the values associated with the key. In this function, various tasks can be performed, such as data aggregation and filtering. Once the function is completed, the results in the key/value pair are sent to the Output function.

The Analytic Engine of the DHAA can be fitted into a MapReduce pattern to enhance analysis efficiency. The enhanced pattern is structured as Figure 15.

The data from different sources are processed independently by the nodes of data alignment. These tasks can be performed in parallel to accelerate the processing, and they should be placed in the Map step of the analysis. The aligned data will be aggregated by the nodes of feature extraction based on their data sources. This grouping operation can be achieved by the key/value pairs generated in the Map step. The processes of feature extraction should be placed in the Map function. The processes of feature integration and target state estimation need to be rendered on the base of the prior two components. Hence, these processes should be placed in the Output function. The pseudocode for the DHAA MapReduce pattern is outlined as shown in Pseudocode 1.

We are implementing a health analytic system based on the above architecture to enhance the health report component of AIA Vitality. We name it Vitality Health Analytic Platform (VHAP, Figure 16). Through the implementation, we can also evaluate the applicability of DHAF form empirics. An experimental system is built on a virtual machine with the Ubuntu Linux operating system and Hadoop, which is a big data-oriented framework that enables distributed storage and distributed processing in computing clusters [51].

```
(1) //set up each data item and encode data in key-value pairs
(2) class mapper:
(3)      setup():
(4)           initialize input element
(5)       map(key,element):
(6)           flag data source to reducer
(7)           identify data type
(8)           normalize data element
(9)       cleanup():
(10)          emit (key,aligned element)
(11) //process key-value pairs and update local model
(12) class reducer:
(13)     setup():
(14)          initialize aligned element
(15)          initialize local situation representations
(16)     reduce(key,elements):
(17)          for element in elements:
(18)               if element meets hypothesis:
(19)                    update local situation representation per flag
(20)     cleanup():
(21)          emit (key,local situation representations)
(22) //adjust global model and perform predictions
(23) class outputter:
(24)     setup():
(25)          initialize input elements
(26)          initialize local situation representations
(27)          initialize global situation representation
(28)     output(key,elements,representations):
(29)       for element in input elements:
(30)          if element has proper local situation representation:
(31)             update global situation
(32)             train neural network
(33)             estimate error
(34)             update neural network weights
(35)          else:
(36)             perform neural network prediction
(37)          perform regression prediction
(38)          generate report
(39)       cleanup():
(40)          emit report
```

Pseudocode 1

The Analytic Engine is implemented with R programming language, which provides various libraries convenient for statistical computing and visualization. RHadoop consists of several libraries that allow easy and tight integration between R and Hadoop. Typically, the "rmr2" library contains a collection of functions for implementing MapReduce processing, the "rdfs" library provides interfaces for accessing to HDFS (Hadoop Distributed Filesystem), and the "rhbase" provides an interface for accessing HBase that is a NoSQL database running on top of Hadoop.

There are also numerous R libraries for implementing machine-learning functions. For building the neural networks used in the components of data alignment and feature integration, we can use the "h2o," "deepnet," and "MXNetR" libraries. In particular, the "MXNetR" package contains interfaces for implementing convolutional neural networks. For implementing the Bayesian rules for hypothesis selection in feature extraction, we can use the "OpenBUGS" library. For constructing the Bayesian Networks used in the components of feature extraction and target state estimation, we can use "bnlearn" and "gRain." For building the regression model for time series analysis in the target state estimation component, we can use the "arima" library. By setting arima$(2, 0, 1)$ in the library, we can perform the ARMA$(2, 1)$ analysis. Some sample R scripts for the implementation of machine learnings are provided in Pseudocode 2 as an indication of implementation.

The Data Governance component in VHAP is mainly implemented with the HDFS and HBase of Hadoop. HDFS is used for storing the configuration of external knowledge and auxiliary information. The advantages of using HDFS include the support for files of large size and volume and

```
(1) ## train an neural network for health risk prediction
(2) trainCNN <- function(trainset,risks) {
(3)    require(mxnet);
(4)    # Define the input data
(5)    data <- mx.symbol.Variable('data');
(6)    # 1st convolutional layer
(7)    conv_1 <- mx.symbol.Convolution(data = data, kernel = c(5,5), num_filter = 20);
(8)    act_1 <- mx.symbol.Activation(data = conv_1, act_type = "relu")
(9)    pool_1 <- mx.symbol.Pooling(data = act_1, pool_type = "max", kernel = c(2,2),
       stride = c(2,2))
(10)   # 2nd convolutional layer
(11)   conv_2 <- mx.symbol.Convolution(data = pool_1, kernel = c(5,5), num_filter = 50)
(12)   act_2 <- mx.symbol.Activation(data = conv_2, act_type = "relu")
(13)   pool_2 <- mx.symbol.Pooling(data=act_2, pool_type = "max", c(2,2), stride = c(2,2))
(14)   # 1st fully connected layer
(15)   flatten <- mx.symbol.Flatten(data = pool_2)
(16)   fc_1 <- mx.symbol.FullyConnected(data = flatten, num_hidden = 100)
(17)   act_3 <- mx.symbol.Activation(data = fc_1, act_type = "relu")
(18)   # 2nd fully connected layer
(19)   fc_2 <- mx.symbol.FullyConnected(data = act_3, num_hidden = 40)
(20)   # Set up the symbolic model
(21)   NN_model <- mx.symbol.SoftmaxOutput(data = fc_2)
(22)   # Set seed for reproducibility
(23)   mx.set.seed(0)
(24)   # Train the model
(25)   model <- mx.model.FeedForward.create(NN_model, X=trainset, y=risks, ctx=mx.cpu(),
       num.round = 50, array.batch.size = 40, learning.rate = 0.01, momentum = 0.9)
(26)   return(model);
(27) }
(1) ## use an neural network to predict health risk
(2) predictRisk <- function(nn,dataset,risks) {
(3)    require(neuralnet);
(4)    nn.results <- compute(nn, dataset);
(5)    pr = creditnet.results$net.result;
(6)    colnames(pr) <- risks;
(7)    return(pr);
(8) }
(1) ## use ARMA to predict health risk and plot
(2) plotARMA <- function(dataset,start=c(2011,1)) {
(3)    require(forecast);
(4)    datat<-ts(dataset[,2],start=start,frequency=12);
(5)    fit <- Arima(datat, order=c(2,0,1));
(6)    plot(forecast(fit));
(7) }
```

Pseudocode 2

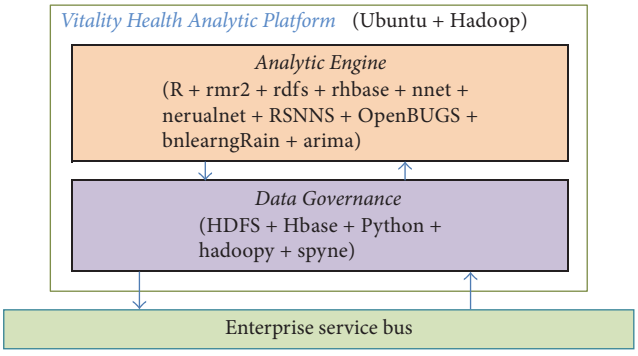


Figure 16: Vitality Health Analytic Platform.

high efficiency for frequent data access. HBase is used for caching the raw data and aligned data, managing the state representations, and keeping the analysis reports. In contrast to traditional databases, HBase provides a fault-tolerant way of storing large quantities of sparse data. The communication between Data Governance and the external system is through the web services implemented by Python, which is a programming language with reliable Hadoop integration and handy network communication supports. The "hadoopy" library enables Python to interact with Hadoop. The "spyne" library provides a framework for implementing and exposing remote procedure-call APIs. These APIs are exposed to the enterprise service bus, and Data Governance becomes

Figure 17: New health report versus old health report.

the interface between the Analytic Engine and the external systems.

In comparison with the previous health reports in AIA Vitality (see Figure 17), the new health reports using VHAP can provide health risk predictions with higher granularity. While the previous health reports based on data rules can only provide health predictions with broadly indicative predictions, the new prediction can give indicators of health risks on an annual basis and can predict future trends for each health risk.

In order to verify the performance of the selection of CNN for fusion cell implementation, we substituted the implementation of CNN with other learning techniques in terms of SVM, KNN, and LR for comparison. The health records of approximately 20,000 AIA Vitality members were sampled for the calculation. The number of the original records of the sample members was about 800,000,000. However, there were many duplicated records which are produced from their source systems. For example, Vitality Mobile App users tend to upload the same data several times during a day. On the other hand, invalid records also exist, likely due to the malfunction of the source systems or the inappropriate operations of the software by users. The data quality needs to be controlled to ensure the accuracy and performance of the analytics. This is achieved by implementing a filter in the data alignment node, which screens out the duplicated and outlier data before further processes are applied. The classification accuracy and kappa measure for the specific implementations are given in Figure 18, and their efficiency measures are given in Figure 19.

From Figure 18, it can be seen that CNN has relatively high accuracy compared to other learning methods. However, during the experiments, we noticed that the process duration of CNN was longer than other methods. We believe that is caused by the complex implementation of CNN. In contrast, KNN is a simpler classification algorithm. While its overall accuracy is acceptable, its accuracy is usually not stabilized depending on the shape of the dataset. LR is even more sensitive to the distribution of the sample data. Outliers in the sample data can dramatically affect the result. Both SVM and CNN are state-of-the-art learning algorithms. They both learn from the inputs and training data by adjusting their weights. SVM achieves the weighting by mapping the inputs to higher dimensional space, while CNN achieves it by producing multilayer feature maps. The complex architecture of CNN makes it a stronger classifier and its results are more reliable.

According to Figure 19, CNN has the longest process duration and lowest throughput when performing the sample analytics, whereas LR and KNN require much less processing time and have high throughput when analyzing the sample data. However, taking into account the accuracy figures, although CNN consumes more process time, it produces higher accuracy. In contrast, LR requires less processing time and its accuracy is also low. In addition, the process duration can be reduced if we increase the capacity of the platform. Nevertheless, it is more difficult to increase the accuracy of analytics than enhance the computing capacity.

6. Conclusions

This research explores the potential value for utilizing the concepts of the fusion node paradigm and deep learning to enhance health risk prediction from big data. The fusion node paradigm improves the structure and workflow of the health analytic design, so that the analytic processes can be conducted in a more efficient way. Deep learning compounds multiple machine-learning methods in complex processes to enhance data analysis performance. Using deep learning techniques in the analytic processes allows health risks to be predicted in a more reliable manner, as it allows iterative inference of high-level information based on low-level data. In addition, deep learning methods such as convolutional neural network have the potential to enhance analysis accuracy with large training sets. This specialism has great merit in this era of big data. A framework is proposed based on

Risks	SVM	KNN	LR	CNN
Obesity	53.89	84.98	35.61	71.43
Diabetes	79.06	65.65	44.87	96.28
Stroke	65.77	67.55	84.56	47.36
Heart disease	73.43	84.09	23.56	72.81
High blood pressure	68.51	52.48	61.13	75.42
Osteoarthritis	77.57	65.31	67.38	75.35
Cancer	60.09	55.46	34.74	80.47
Overall	67.41	65.32	48.86	73.21
Kappa	0.59	0.57	0.41	0.63

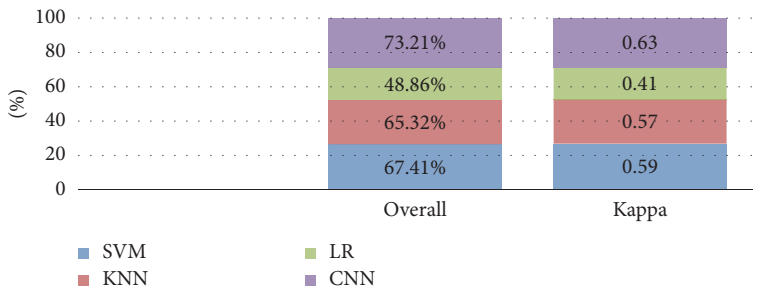


Figure 18: Accuracy and kappa of various implementations.

	SVM	KNN	LR	CNN
Proc duration (Min)	1247	893	694	1652
Throughput (rows/Sec)	4147	5725	7359	3202
Throughput (KB/Sec)	3390	4644	5944	2610

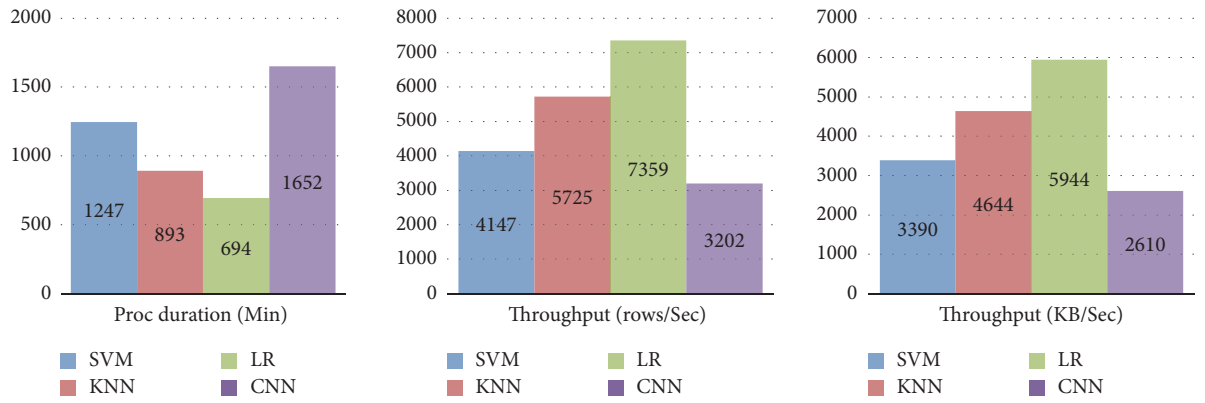


Figure 19: Efficiency of various implementations.

the concepts described above to discuss the theories and processes of the analytic workflow and its components. An architecture is designed based on the proposed framework to illustrate how to organize the analytic components of the analytic system to capitalize on the cohesion and coupling. This paper also discussed the application of the MapReduce paradigm to optimize the efficiency of the analytic processes in the architecture. Finally, we encourage SOA design of the analytic system to ease the ability of external systems to access the analytic results.

In the future, we plan to perform more analysis on the performance of the proposed prediction framework. More observations are required to verify whether the predicted probabilities are aligned with the actual occurrence rates. Besides the accuracy, computation efficiency is another concern of analytic systems. There need to be an observation of the process duration and resource consumption and a comparison of different algorithms in regard to the analytic processes. We can also attempt to devise different mathematic models for probability aggregation.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] J. Wang, Z.-Q. Zhao, X. Hu, Y.-M. Cheung, H. Hu, and F. Gu, "Online learning towards big data analysis in health informatics," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8211, pp. 516–523, 2013.

[2] W. Liu and E. Park, "Big data as an e-health service," in *Proceedings of the 2014 International Conference on Computing, Networking and Communications ICNC 2014*, pp. 982–988, February 2014.

[3] E. Mezghani, E. Exposito, K. Drira, M. Da Silveira, and C. Pruski, "A Semantic Big Data Platform for Integrating Heterogeneous Wearable Data in Healthcare," *Journal of Medical Systems*, vol. 39, no. 12, article no. 185, 2015.

[4] L. Deng and D. Yu, "Deep learning: Methods and applications," *Foundations and Trends in Signal Processing*, vol. 7, no. 3-4, pp. 197–387, 2013.

[5] H. Chai and B. Wang, "A hierarchical situation assessment model based on fuzzy Bayesian network," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7003, no. 2, pp. 444–454, 2011.

[6] Y. Wang, L. Kung, and T. A. Byrd, "Big data analytics: understanding its capabilities and potential benefits for healthcare organizations," *Technological Forecasting & Social Change*, pp. 1–11, 2016.

[7] N. Jothi, N. A. Rashid, and W. Husain, "Data Mining in Healthcare - A Review," in *Proceedings of the 3rd Information Systems International Conference*, pp. 306–313, April 2015.

[8] S. Theodoridis, *Machine Learning - A Bayesian and Optimation Perspective*, Elsevier, 2015.

[9] S. Zillner and S. Neururer, "Big data in the health sector," in *New Horizons for a Data-Driven Economy*, pp. 179–194, Springer International Publishing.

[10] I. Goodfellow and A. Courville, *Deep learning*, MIT Press, 2016.

[11] Z. Liang, G. Zhang, J. X. Huang, and Q. V. Hu, "Deep learning for healthcare decision making with EMRs," in *Proceedings of the IEEE International Conference on Bioinformatics and Biomedicine*, pp. 556–559, November 2014.

[12] E. Bosse, J. Roy, and S. Wark, *Concepts, Models, and Tools for Information Fusion*, Artech House, 2007.

[13] X. Gao, W. Lia, M. Loomesa, and L. Wang, "A fused deep learning architecture for viewpoint classification of echocardiography," *Information Fusion*, vol. 36, pp. 103–113, 2017.

[14] NIST Big Data Public Working Group, "NIST Big Data Interoperability Framework," 2015, http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.1500-4.pdf.

[15] Y. Wang, L. Kung, C. Ting, and T. A. Byrd, "Beyond a technical perspective: Understanding big data capabilities in health care," in *Proceedings of the 48th Hawaii International Conference on System Sciences*, pp. 3044–3053, January 2015.

[16] B. Lantz, *Machine Learning with R 2ed*, Packt Publishing, 2015.

[17] J. Bell, *Machine Learning - Hands-On for Developers and Technical Professionals*, John Wiley & Sons, 2015.

[18] E. Vermeulen-Smit, M. Ten Have, M. Van Laar, and R. De Graaf, "Clustering of health risk behaviours and the relationship with mental disorders," *Journal of Affective Disorders*, vol. 171, pp. 111–119, 2015.

[19] D. Talia, D. Talia, and F. Marozzo, *Data Analysis in the Cloud - Models, Techniques and Applications*, Elsevier Inc, 2016.

[20] J. Kazmierska and J. Malicki, "Application of the Naïve Bayesian Classifier to optimize treatment decisions," *Radiotherapy and Oncology*, vol. 86, no. 2, pp. 211–216, 2008.

[21] P. Harrington, "Machine learning in action," *Mainning Publications*, 2012.

[22] J. Sander and J. Beyerer, "Bayesian fusion: Modeling and application," in *Proceedings of the Sensor Data Fusion: Trends, Solutions, Applications*, pp. 1–6, October 2013.

[23] L. A. Klein, *Sensor and Data Fusion: A Tool for Information Assessment and Decision Making*, SPIE Press, 2004.

[24] A. Gelman, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, *Bayesian Data Analysis*, CRC Press, 3rd edition, 2014.

[25] N. Fenton and M. Neil, "Decision support software for probabilistic risk assessment using bayesian networks," *IEEE Software*, vol. 31, no. 2, pp. 21–26, 2014.

[26] M. L. Vaneek, "Introduction into Bayesian Networks," 2016, http://www.fit.vutbr.cz/study/courses/VPD/public/0809VPD-Vanek.pdf.

[27] D. Barber, *Bayesian Reasoning and Machine Learning*, Cambridge University Press, 2012.

[28] D. E. Holmes and L. C. Jain, *Innovations in Bayesian Networks*, Springer, 2008.

[29] A. Esposito, S. Bassis, and F. C. Morabito, "Recent advances of neural networks models and applications: An introduction," *Smart Innovation, Systems and Technologies*, vol. 37, pp. 3–8, 2015.

[30] S. Samarasinghe, *Neural Network for Applied Sciences and Engineering*, Auerbach Publications, 2007.

[31] M. Kirk, *Thoughtful Machine Learning*, O'Reilly Media.

[32] N. Lewis, *Deep Learning Made Easy with R*, CreateSpace, 2016.

[33] N. Dhungela, G. Carneirob, and A. P. Bradley, "A deep learning approach for the analysis of masses in mammograms with minimal user intervention," *Medical Image Analysis*, vol. 37, pp. 114–128, 2017.

[34] M. Pereza, S. Avilab, D. Moreiraa et al. et al., "Video pornography detection through deep learning techniques and motion information," *Neurocomputing*, vol. 230, pp. 279–293, 2017.

[35] Q. Zhang, L. T. Yang, and Z. Chen, "Privacy preserving deep computation model on cloud for big data feature learning," *Institute of Electrical and Electronics Engineers. Transactions on Computers*, vol. 65, no. 5, pp. 1351–1362, 2016.

[36] S. Das, *High-Level Data Fusion*, Artech House, 2008.

[37] M. K. Sundareshan and Y. C. Wong, "Nueral network-based fusion of image and non-image data for surveillance and tracking application," Data Fusion for Situation Monitoring, Incident Detection, Alert and Response Management, no. IOS Press, 2015.

[38] H. B. Mitchell, "Data fusion: Concepts and ideas," *Data Fusion: Concepts and Ideas*, pp. 1–344, 2012.

[39] D. L. Hall and J. Llinas, *Handbook of Multisensor Data Fusion*, CRC Press, 2001.

[40] E. Bosse and B. Solaiman, *Information Fusion and Analytics for Big Data and IoT*, Artech House, 2016.

[41] L. Wald, *Data Fusion: Definitions and Architectures*, Les Presses de L'Ecole des Mines, 2002.

[42] M. C. Amirani, M. Toorani, and A. A. Beheshti, "A new approach to content-based file type detection," in *Proceedings of the 13th IEEE Symposium on Computers and Communications, ISCC 2008*, pp. 1103–1108, July 2008.

[43] D. Tian, A. Gledson, A. Antoniades, A. Aristodimou, and N. Dimitrios, "A bayesian association rule mining algorithm," in *Proceedings of the IEEE International Conference on Systems*, pp. 3258–3264, October 2013.

[44] D. Allard, A. Comunian, and P. Renard, "Probability aggregation methods in geoscience," *Mathematical Geosciences*, vol. 44, no. 5, pp. 545–581, 2012.

[45] O. Abdel-Hamid, A.-R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 22, no. 10, pp. 1533–1545, 2014.

[46] M. Sun, Z. Song, X. Jiang, J. Pan, and Y. Pang, "Learning Pooling for Convolutional Neural Network," *Neurocomputing*, vol. 224, pp. 96–104, 2017.

[47] Q. Li, Z. Jin, C. Wang, and D. D. Zeng, "Mining opinion summarizations using convolutional neural networks in Chinese microblogging systems," *Knowledge-Based Systems*, vol. 107, pp. 289–300, 2016.

[48] R. Prado and M. West, *Time Series - Modeling, Computation, and Inference*, CRC Press, 2010.

[49] D. Miner and A. Shook, *MapReduce Design Patterns*, O'Reilly Media, 2013.

[50] B. Holt, *Writing and Querying MapReduce Views in CouchDB*, O'Reilly Media, 2011.

[51] V. Prajapati, *Big Data Analytics with R and Hadoop*, Packt Publishing, 2013.

Hindawi
Scientific Programming
Volume 2017, Article ID 2573592, 11 pages
https://doi.org/10.1155/2017/2573592

Research Article

An Association-Oriented Partitioning Approach for Streaming Graph Query

Yun Hao,[1,2,3] Gaofeng Li,[1,2,3] Pingpeng Yuan,[1,2,3] Hai Jin,[1,2,3] and Xiaofeng Ding[1,2,3]

[1]*Services Computing Technology and System Lab., School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China*
[2]*Cluster and Grid Computing Lab., School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China*
[3]*Big Data Technology and System Lab., School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China*

Correspondence should be addressed to Pingpeng Yuan; ppyuan@mail.hust.edu.cn and Hai Jin; hjin@hust.edu.cn

Received 26 December 2016; Accepted 11 April 2017; Published 25 May 2017

Academic Editor: Alex M. Kuo

The volumes of real-world graphs like knowledge graph are increasing rapidly, which makes streaming graph processing a hot research area. Processing graphs in streaming setting poses significant challenges from different perspectives, among which graph partitioning method plays a key role. Regarding graph query, a well-designed partitioning method is essential for achieving better performance. Existing offline graph partitioning methods often require full knowledge of the graph, which is not possible during streaming graph processing. In order to handle this problem, we propose an association-oriented streaming graph partitioning method named Assc. This approach first computes the rank values of vertices with a hybrid approximate PageRank algorithm. After splitting these vertices with an adapted variant affinity propagation algorithm, the process order on vertices in the sliding window can be determined. Finally, according to the *level* of these vertices and their association, the partition where the vertices should be distributed is decided. We compare its performance with a set of streaming graph partition methods and METIS, a widely adopted offline approach. The results show that our solution can partition graphs with hundreds of millions of vertices in streaming setting on a large collection of graph datasets and our approach outperforms other graph partitioning methods.

1. Introduction

With the rapid development of Internet, huge amounts of graph data are emerged every day. For example, the Linked Open Data Project which aims to connect data across the web published 149 billion triples till 2017 while 31 billion triples were published in September 2011 [1]. In addition, dynamic real-world graphs include social networks. The dynamicity of graphs stems from either the world-wide hot events or updates of the web contents [2]. Thus, the rapid explosion of data volume and dynamicity urgently necessitates large scale graph analysis applications which can handle these dynamic workloads. To achieve a desirable performance in big dynamic graph analysis, streaming graph partitioning algorithms, which give a guarantee of scalability by distributing a streaming graph to multiple machines, will be employed inevitably.

In fact, graph partitioning problem has received a lot of attention over the last years. The existing algorithms may be grouped in two divisions: edge cut algorithms and vertex-cut algorithms. The majority of distributed graph engines adopt edge-based hash partitioning [3–6] as the data partitioning solution. Edge-based hash partitioning is a vertex-cut approach which distributes edges across the partitions by computing the hash keys of vertices and allows edges to span partitions. For example, Pregel [7] uses a special hash function to distribute the vertex. The principle of this approach is to collect those edges which share the same vertex and distribute the vertices to different computing nodes. Hence, this approach can obtain good performance for simple graph operations, such as answering star queries. Although hash approach generates a balanced number of vertices across distributed computing nodes, it entirely ignores the graph structure. As a result, many messages have to be sent across

the nodes when executing graph operations. It leads to heavy communication traffic. All in all, this approach has extremely poor locality, which means that the complex graph query operations may incur frequent cross-node interactions and intermediate result exchanging. Consequently, the benefit of distributed and parallel processing of the big graph data is lost or significantly reduced due to the high communication overhead incurred by hash partitioning scheme. Wu et al. proposed path partitioning for scalable SPARQL query processing over static RDF graphs [8]. Path partitioning approach does not divide a graph into a set of independent edges or vertices but sets of paths. Thus, the approach can largely reduce the cost of distributed joins over the large RDF dataset.

In this paper, we propose an association-oriented partitioning approach for streaming graphs. Our approach is based on one important observation: in order to minimize the interactions among partitions, we need to consider the associations among vertices when we assign vertices and edges to partitions. The main contributions of the paper are twofold. *First*, for a streaming situation in which large scale graph data arrives fast and continuously, we propose an association-oriented partitioning approach. The approach considers the association among recent arriving graph data which falls in a sliding window. The approach first computes rank scores of vertices in the sliding window and then clusters the vertices and edges according to the rank values of vertices and associations between vertices. By partitioning the big graphs into multiple partitions with this approach, we reduce interactions among partitions and the cost for internode communication. *Second*, we evaluate our approach in labeled and unlabeled streaming graphs through extensive experiments. The experimental results show that our approach outperforms HASH approach and METIS [9]. The reason is that our approach reduces the interactions between partitions. The results also show that our approach is capable of handling incrementally generated streaming data.

2. Preliminary

In this section, we will present the concepts used in the paper. Our approach can handle both directed and undirected graphs. In addition, labeled and unlabeled graphs can be processed, too. Since undirected graph can be easily transformed into directed graph by adding another edge between two connected vertices, the following discussion mainly focuses on directed connected graph defined in [5, 6].

Generally, the edges or vertices, or both, of a graph are assigned labels. A graph with labeled vertices is named as a vertex-labeled graph. Similarly, in an edge-labeled graph each edge has a label. In a directed edge-labeled graph, the label of an edge indicates the relationship between its source vertex and target vertex. An edge with its two vertices ($a \xrightarrow{b} c$) can be represented as a triple (a, b, c). In RDF (Resource Description Framework) graph (e.g., Figure 1(b), a fragment from LUBM [10]), a is called subject and c is the object of the triple. Label b on the edge is the predicate of the triple. In the following, labeled graph will be defined formally.

Definition 1 (association of vertices). Regarding a graph $G(V, E)$, the association between handling u and v is defined as $A(u, v)$; min(mvn) is the minimal number of vertices on a path that connects the two vertices:

$$A(u,v) = \begin{cases} 1 & \text{directly connected} \\ \dfrac{1}{\min(\text{mvn}) + 1} & \text{indirectly connected} \\ 0 & \text{unconnected.} \end{cases} \quad (1)$$

A graph partitioner assigns vertices or edges to k partitions. The associations of the partitions should be low in order to reduce the interactions among partitions. Here we define the association of two clusters.

Definition 2 (association of clusters). Assume two clusters (partitions) C_a and C_b. e_a, e_b are the exemplars of clusters C_a, C_b. Exemplar is found by randomly choosing an initial subset of data points and then iteratively refining it. The association $A(C_a, C_b)$ between cluster C_a and cluster C_b is defined as follows:

$$A(C_a, C_b) = \begin{cases} A(e_a, e_b) & e_a, e_b \text{ are connected} \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

Graph partitioning approach should distribute vertices to each cluster uniformly. It is also critical that the total number of cross-partition edges is small, in order to minimize the communication cost between different partitions.

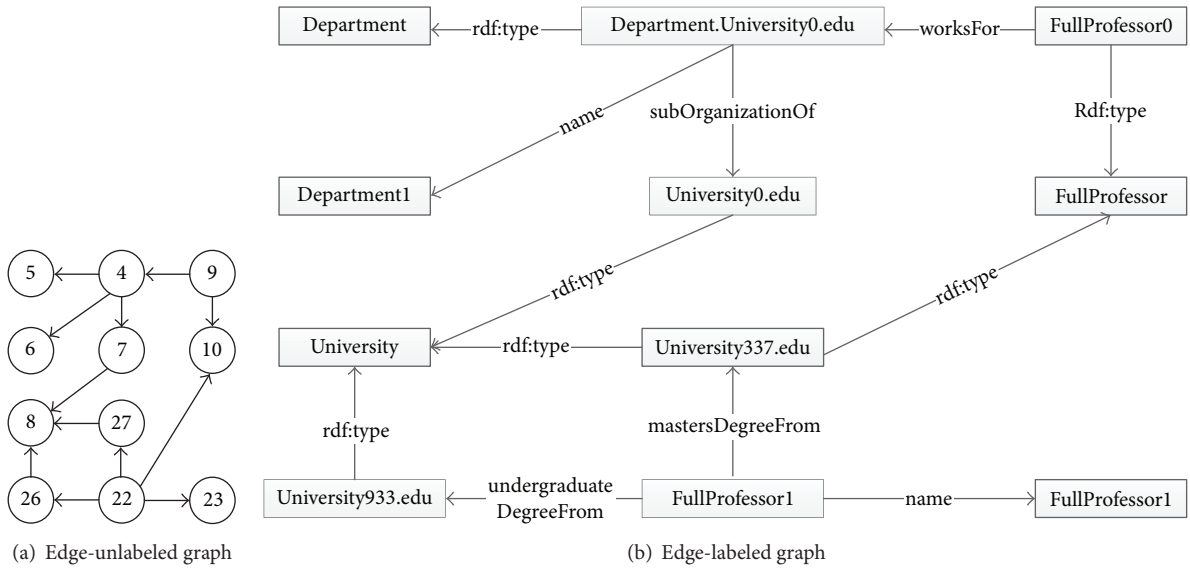
Definition 3 (edge cut). Assume graph $G(V, E)$, where V is the set of vertexes in the graph and E is the set of edges in the graph. Assume a partition of G consists of $P_1, P_2, \ldots, P_n$. Namely, $P_1 \cup P_2 \cup \cdots \cup P_n = G$. For partition P_i, $\forall v \in P_i$,

$$\text{ec}(v) = \begin{cases} 1 & ((v,t) \in G \vee (t,v) \in G) \wedge t \notin P_i \\ 0 & \text{otherwise;} \end{cases} \quad (3)$$

the edge cut of P_i is $\text{ec}(P_i) = \sum_{v \in P_i} \text{ec}(v)$. The edge cut of the partition is $\sum_{0 \le i < n} \text{ec}(P_i)$.

Minimizing the number of *edge cuts* may not be the only goal for partitioning approaches, as the cost of processing graphs is determined by not only the network communication but also the size of the messages, although each message typically is small and contains a little information. So it is not enough to minimize the total number of edges crossing different partitions. Minimizing the *communication volume* across computing nodes is also a goal.

Definition 4 (communication volume). Assume $P_1, P_2, \ldots, P_n$ is a partition of graph $G(V, E)$. $P_1 \cup P_2 \cup \cdots \cup P_n = G$. Communication volume $\text{cv}(P)$ is positive correlation with edge cut.



(a) Edge-unlabeled graph (b) Edge-labeled graph

Figure 1: Example of unlabeled and labeled graph.

3. Association-Oriented Streaming Graph Partitioning

In this section, we present our association-oriented approach for streaming graph partitioning. Streaming graph partitioning tries to distribute nodes and edges into multiple machines, while it should keep the data balance and the communication volume minimal. The streaming partitioner receives graph data and then decides where the nodes of the graph data should be distributed. Our approach first orders each vertex in partitions and then computes the association between a newly arrived vertex and other vertices which are distributed in a partition. Then we follow the rank values of vertices to merge the newly arrived vertices with other vertices. The rank values of vertices indicate the step in which vertices will be merged with others. At last, we use a variant of affinity propagation to cluster nodes. Thus, it faces two challenges. The first challenge is how to order the vertex that needs to be merged in the merging steps, as it may lead to extremely skewed data distribution if the order is not good enough. At the same time, both space and time complexity of association partitioning are high when the number of merging steps is large. More merged vertex means fewer replicas and higher intra-association. However, it leads to a skew in data distribution since there may be some very large clusters, which are undesirable. Consequently, we present an approach to keep the data distribution balanced. These techniques are discussed in terms of the light-weight AP-based stream graph clustering and association partitioning algorithm in following. More importantly, the processor will use some optimization strategies such as using start vertices instead of all vertices in the clustering process.

3.1. Hybrid Approximate PageRank Computation. Since the association merging is to merge vertices in each step, we need to order vertices first. The reason is to enable the vertices with a higher score to be merged later, because merging these vertices first may result in very large clusters. PageRank score of vertices can be used to rank vertices. At the same time, we have to group similar vertices by their PageRank score in order to simplify the computation. More specifically, for vertex v, we use a term *Level* (L) to indicate the level of the start vertices of each cluster that has the maximum association with vertex v, denoted by $L(v) = n$, $n \in \mathbb{N}^+$. Generally, the level starts from 1 and the level of the root is 1. Here, since our approach merges vertices from down to top, the level of leaves which is the deepest is 1. Namely, the level of a node is defined by the depth of the tree minus the number of connections between the node and the root. Sets of vertices which have similar PageRank values may have the same L value n, which exactly will be merged at the step n. By using L as the criterion for merging clusters, the number of steps in processing association partitioning can be reduced substantially.

The vertex rank-based grouping approach is to order the vertices in ascending order by their PageRank score. The PageRank for a directed graph can be seen as a way of ranking the entire distribution of the degree of each vertex. We first calculate the PageRank score for each vertex.

An important point to note is that it is not possible to access all the information in a streaming graph, in particular when the data are arriving continuously; that is, the volume of data is increasing. On the other hand, since arrived data has been distributed to storage nodes, it is difficult to analyze all of the data due to its excessive volume as well as the cost for data transmission. It is desirable if we can estimate the PageRank of some selected nodes quickly.

To address this problem, we approximate the PageRank of each vertex from its in-degree [11]. Equation (4) shows that the average PageRank of nodes k is proportional to its in-degree:

$$\overline{p}(k) = \frac{d}{N} + \frac{1-d}{N}\frac{k_{\text{in}}}{\langle k_{\text{in}} \rangle}. \quad (4)$$

Table 1: Proportion of each predicate in LUBM.

Predicate	Type	Name	teacherOf	worksFor	memberOf	Advisor
Proportion	20.0402%	15.4818%	1.5641%	0.5209%	7.5539%	2.9666%

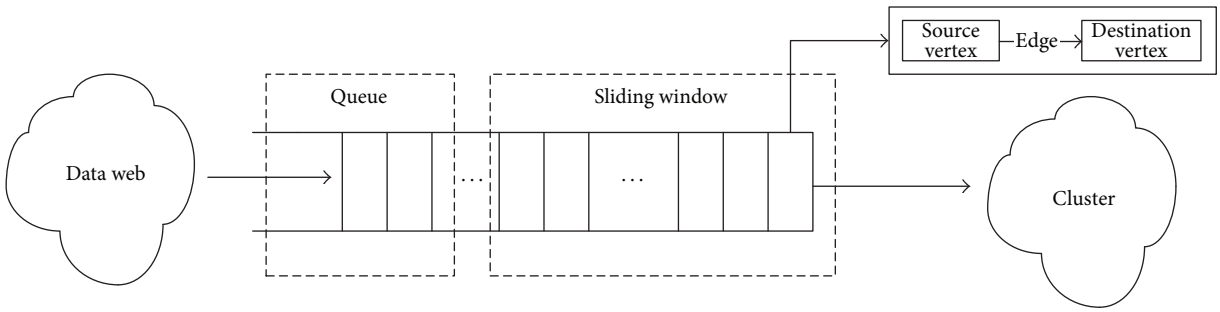


Figure 2: Sliding window.

Generally, PageRank algorithm does not consider the difference between edges. However, in labeled graphs, they may be different, which requires considering the edge weight. For example, in Figure 1(b), each triple in the RDF graph can be treated as a direct edge whose subject as a source points to object as a destination and the predicate as a label for the edge. The proportion of same predicates in triples is different. Experimental results show that the total number of predicates in LUBM [10, 12] is 18, and the selected six predicates in Table 1 show that each p_i has a relatively stable proportion to the total in a data set.

Suppose edge points from s_{sub} to o_{obj} with label p_i; the PageRank of o_{obj} contributed from s_{sub} is computed as

$$\text{PR}\left(s_{\text{sub}} \longrightarrow o_{\text{obj}}\right) = \frac{w_{p_i}}{\sum_{\forall s: s \xrightarrow{p} o_{\text{obj}}} w_p} \text{PR}\left(s_{\text{sub}}\right), \tag{5}$$

where w_{p_i} is the percentage of p_i in total number of p:

$$w_{p_i} = \frac{\left|p_i\right|}{\sum_{j<\|P\|} \left|p_j\right|}, \tag{6}$$

and the sum of w_{p_j} is the total percentage of p that s_{sub} has. The quotient means the weight of p_i assigned by the subject s_{sub}. Multiplication by PageRank of s_{sub} is the contribution that s_{sub} gives to o_{obj}.

Then a hybrid approximate algorithm for computing the PageRank score quickly in the sliding window is as follows:

$$\text{PR}_{o_{\text{obj}}} = \alpha \text{PR}_{\text{InDeg}} + (1-\alpha) * \text{PR}_{w_p}, \tag{7}$$

where PR_{InDeg} is the PageRank value approximated by the in-degree value of vertex o_{obj} and PR_{w_p} is the total contribution that comes from all s_{sub} pointing to o_{obj}. $\alpha \in (0, 1)$. α is a hybrid factor that needs to be trained by some heuristic algorithm like simulated annealing. Using (4) and (5) we can get the following expression. With the expression, we can compute PageRank values more quickly:

$$\begin{aligned} \text{PR}_{o_{\text{obj}}} = {} & \alpha \left[\frac{d}{N} + \frac{1-d}{N} \frac{k_{\text{in}}}{\langle k_{\text{in}} \rangle}\right] + (1-\alpha) \\ & * \left[\frac{w_{p_i}}{\sum_{j: j \in s_k \to \forall p} w_{p_j}} \text{PR}\left(s_{\text{sub}}\right)\right]. \end{aligned} \tag{8}$$

3.2. Sliding Window Based Streaming Model. The original implementation of graph partitioner partitions graphs, which are stored in local files, into several parts. The vertices of graph G arrive in a stream with the set of edges. As vertices arrive, a partitioner decides to place the vertex on one computer of multiple machines. The partitioner can distribute the nodes randomly. The hash partitioning may lead to data skew and bad locality. If the partitioner can scan many nodes of the streaming graphs, it can process nodes in a batch and place nodes with closer relationship together. Inspired by this, we extend the model by implementing a buffer so that the partitioner may decide to place any node in the buffer, rather than the one at the front of the stream. The buffer is named as window (Figure 2). The window determines the boundary of the nodes that partitioner can process now. Once the nodes in the window are processed, the window slides by the window size. The sliding window ensures that partitioner processes nodes in a batch, instead of one node each time. Furthermore, in order to give the partitioner more flexibility, we allow it to access to the statistics of previous entire partitions.

3.3. Computing the Association between Vertices. For each vertex v in the sliding window, Algorithm 1 searches all start vertices in the sliding window which v can be reachable from, denoted by $S(v)$. And for each start vertex $u \in S(v)$, we compute the association value $A(u, v)$ for partitioning and denote the maximum start vertex set with $A(u, v)$ for vertex v as $AS(v) = \{u\}$. All the edges of a particular vertex are inserted into the sliding window at random order. Particularly, G_S means the subgraph of graph G made up by all vertices in

Input:
subgraph formed through sliding window: $G_S = (V_S, E_S)$
Output:
start vertex set for each vertex $\{S(v) \mid v \in V_S\}$,
Association set $\{AS(v) \mid v \in V_S\}$
(1) startVertexSet $\{sv_1, \ldots, sv_m\}$
(2) *Queue* $\leftarrow \{sv_1, \ldots, sv_m\}$
while Queue is not empty **do**
(3) $v \leftarrow$ Queue.front(); Queue.pop();
(4) **if** $v \in$ *startVertexSet* **do**
(5) $S(v) \leftarrow v$; $AS(v) \leftarrow v$;
(6) **foreach** $r \in e = (v, r)$
(7) **if** local indegree of $r > 0$ and local outdegree of $r > 0$
(8) Queue.push(r);
(9) **else**
(10) **foreach** $p \in S(u)$ $(u \in \text{edge}(u, v))$, $q \in AS(v)$ **do**
(11) **if** $A(p, v) > A(q, v)$ **do**
(12) **clean** $AS(v)$, $AS(v) \leftarrow \{p\}$;
(13) **else** $A(p, v) = A(q, v)$ **do**
(14) **update** $AS(v)$, $AS(v) \leftarrow \{p\}$;
(15) **if** $sv_{C_i} \in AS(u)$ and $sv_{C_j} \in AS(u)$ **do**
(16) $C_i' \leftarrow C_i \cup C_j$;

Algorithm 1: Retracing start vertex and computing association.

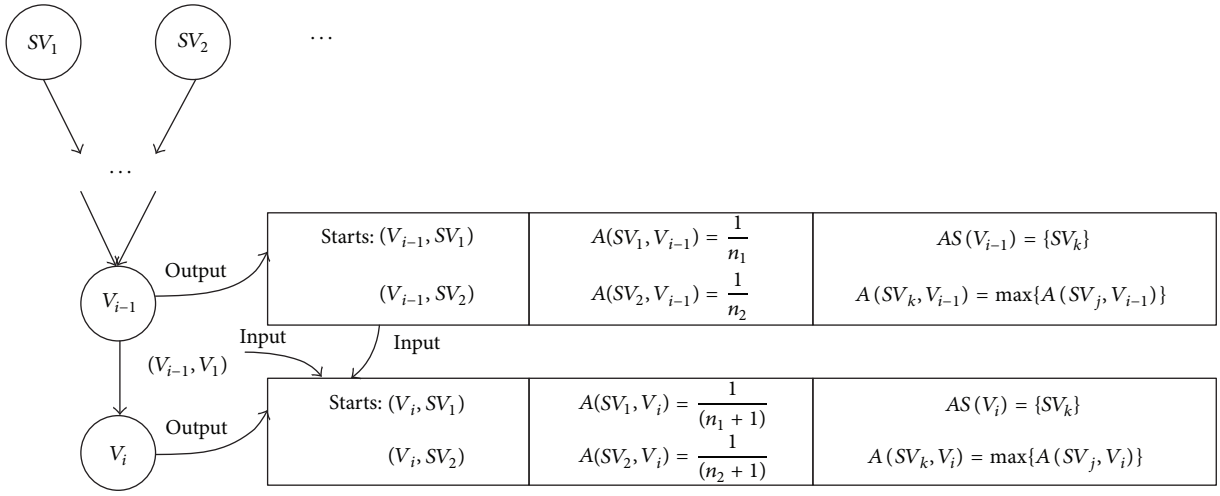


Figure 3: Search start vertices and computing association.

the sliding window. V_S means all vertices in G_S, E_S means all edges in G_S, and sv_{c_i} means the start vertex of cluster c_i.

This algorithm is implemented by iteratively traversing each vertex, which is similar to the breadth-first traversal from each start vertex. Specifically, in ith iteration, it processes the vertices which are an i-hop away from start vertices and gets the start vertex set from the predecessor of these vertices. It then computes $A(u, v)$ and updates AS according to start vertices. For example, in Figure 3, start vertex set of v_{i-1} is $\{sv_1, sv_2\}$, $A(sv_1, v_{i-1}) = 1/n_1$, and $A(sv_2, v_{i-1}) = 1/n_2$. Suppose $A(sv_1, v_{i-1}) = A(sv_2, v_{i-1})$; then $AS(v_{i-1}) = \{sv_1, sv_2\}$. Vertex v_{i-1} is the predecessor of vertex v_i, and by definition of *Association*, $A(sv_1, v_i) = 1/(n_1 + 1)$ and $A(sv_2, v_i) = 1/(n_2 + 1)$. The details are described in Algorithm 1.

3.4. AP-Based Streaming Graph Clustering. After the PageRank is computed, the vertices are divided into multiple subsets by their PageRank score, each of which contains the vertices that share similar PageRank scores. We use modified affinity propagation to cluster nodes. Affinity propagation [13] is an algorithm that takes input measures of similarity between pairs of data points and simultaneously considers all data points as potential exemplars. Unlike the clustering algorithms such as k-means or k-medoids, the number of clusters is not indicated for AP algorithm. AP algorithm also finds exemplars which represent the clusters contained in the input data, while k-medoids does the same thing. AP adopted a message passing algorithm which is called akin belief propagation. Messages are exchanged between data points until a high-quality set of exemplars and corresponding

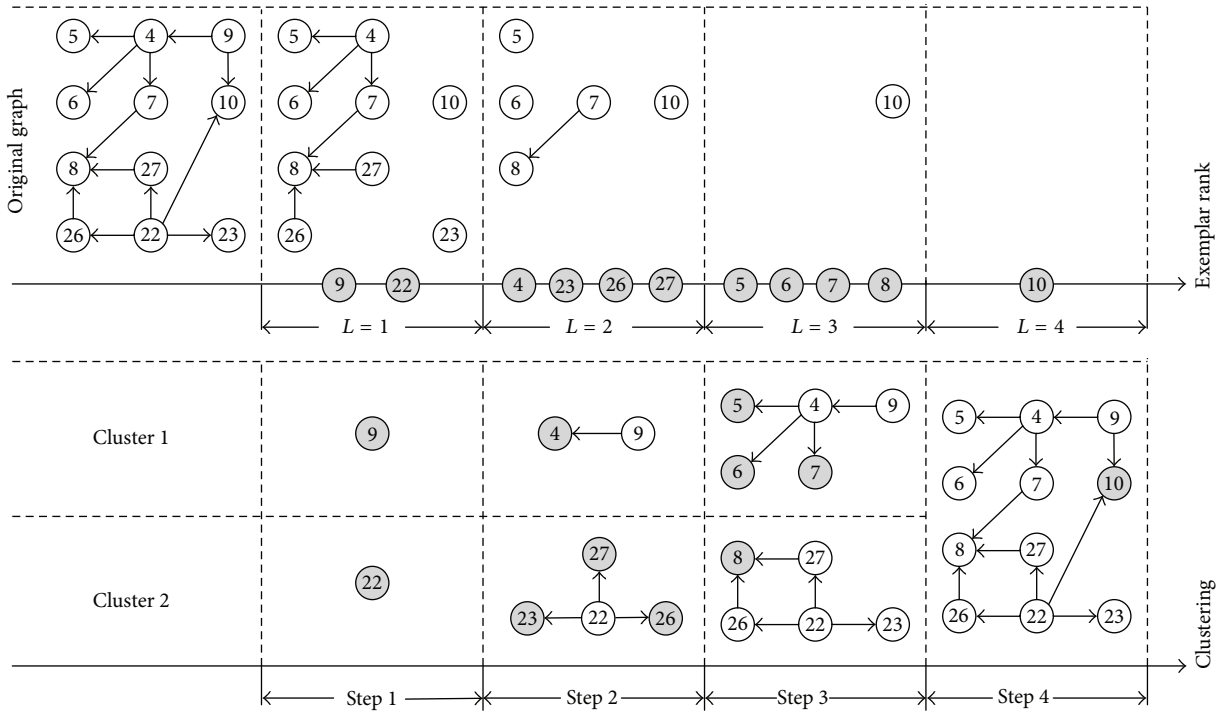


Figure 4: Association clustering.

clusters gradually emerges. If a maximal number of iterations are reached or the exemplars vary little within a fixed number of iterations, the algorithm will stop.

Here, the input data points for this algorithm are the PageRank scores of all the vertices in the sliding window. We use the Euclidean distance as the distance metric between data points, that is, the absolute value of their numerical difference. The affinity propagation algorithm then processes the data by alternating message between data points iteratively until a high-quality set of exemplars and corresponding clusters gradually emerges. Assume that the set of exemplars $\{\text{ex}_1, \ldots, \text{ex}_n\}$ is in a monotonically increasing order by PageRank score; that is, $\text{PR}(\text{ex}_1) < \cdots < \text{PR}(\text{ex}_m)$; then the L values for each exemplar are $L(\text{ex}_1) = 1 < \cdots < L(\text{ex}_n) = m$. The L values of data points in the corresponding cluster are the same as their exemplar. An example is given in Figure 4; $L(4) = L(23) = L(26) = L(27) = 2$ and $\text{ex}_2 = 4$. Algorithm 2 shows the association clustering implementation.

To reduce the overhead of memory and storage and speed up iterations, we simplify the association clustering into start vertices clustering. Note that the start vertices have a minimum L value in vertex ordering strategies; that is, $L(startvertex) = 1$. Based on this, a start vertex can stand for a cluster which contains all vertices at initial phase; for example, vertex 9 can stand for a cluster that merges all vertices with maximum *Association* from vertex 9, as shown in Figure 4.

Specifically, let R_n denote the clustering results at step n. Since a start vertex can stand for the corresponding cluster at initialization, obviously, R_0 is the clustering result at algorithm beginning and each start vertex is a cluster, respectively. The association clustering algorithm then iteratively groups the start vertices step by step. In the iteration at the nth step, for each pair of clusters in R_{n-1}, if the start vertices of these two clusters have the same value of *Association* with vertex u and $L(u) = n$, they will be merged into R_n. This continues until any two sets in R_n are disjoint. For example, in Figure 4, the initial clustering result is $R_0 = \{\{9\}, \{22\}\}$. Since $L(9) = L(22) = 1$, $AS(9) = \{9\}$, and $AS(22) = \{22\}$, then $R_1 = \{\{9\}, \{22\}\}$. Suppose that $L(4) = L(23) = L(26) = L(27) = 2$, $L(5) = L(6) = L(7) = L(8) = 3$, and $L(10) = 4$. At step_1, $R_2 = R_1 \cup \{AS(4), AS(23), AS(26), AS(27)\} = \{\{9\}, \{22\}, \{9\}, \{22\}, \{22\}, \{22\}\}$. We convert R_2 into a pairwise disjoint set; then $R_2 = \{\{9\}, \{22\}\}$. In similar way, $R_3 = \{\{9\}, \{22\}\}$. At step_3, $AS(10) = \{9, 22\}$, $R_4 = \{\{9\}, \{22\}, \{9, 22\}\}$. Finally, $R_4 = \{\{9, 22\}\}$.

As the iterations continue, the termination conditions are a potential problem. In order to generate the final results with acceptable redundancy, high intra-association, and well-balanced distribution, the strategy for the choice of termination conditions has to concentrate on two criteria: the number of clusters remaining (i.e., $|R_n|$ at step n) and the number of steps processed. Obviously, the more the steps are processed, the less the replicas and the higher the intra-associated partitioning results will be produced, but the fewer the number of clusters that will be remaining. Usually, we expect that the number of clusters remaining is larger than the number of storage nodes, for example, λ times compared with the number of storage nodes t.

After performing association clustering, we will assign each cluster to one partition, concentrating on the criterion which produces well-balanced data distribution. Meanwhile, the number of iterations should be as large as possible. In conclusion, the principle of determining the termination

Input:
subgraph formed through sliding window: $sG = (sV, sE)$
Output:
a k-way clustering result $\{P_1, \ldots, P_k\}$
(1) $R_0 \leftarrow \{\{sv_1\}, \{sv_2\}, \ldots, \{sv_m\}\}$, where $sv_i \in SV_{sG}$;
(2) $n \leftarrow 0$;
(3) **repeat**
(4) $n \leftarrow n + 1$;
(5) $R_n \leftarrow R_{n-1} \cup \{AS(u) \mid L(u) = n,\ u \in \{sV - sV_{\text{start}}\}\}$;
(6) **foreach** pair of $C_i, Clu_j \in R_n, C_i \neq C_j$ **do**
(7) **if** $sv_{C_i} \in AS(u)$ and $sv_{C_j} \in AS(u)$ **do**
(8) $C_i' \leftarrow C_i \cup C_j$;
(9) $R_n \leftarrow R_n \cup C_i'$;
(10) $R_n \leftarrow R_n - \{C_i\} - \{C_j\}$;
(11) **until** $|R_n| \leq \lambda k$ or $n > L_{\max}$;
(12) **foreach** $C_i \in R_{n-1}$ **do**
(13) $C_i \leftarrow \text{ExtendByAssociation}(C_i)$;
(14) **foreach** C_i **do**
(15) $P_j \leftarrow \text{DataDistributing}(C_i)$;

Algorithm 2: Association based clustering.

conditions is to merge as many clusters as possible on the basis of keeping the number of remaining clusters larger than times of the number of storage nodes. Hence, the problem can then be formulated as

$$\begin{aligned} \min \quad & \{n\} \\ \text{Subject to} \quad & \left|R_n\right| \leq \lambda t. \end{aligned} \tag{9}$$

Then, the iteration of association clustering will stop at step $n - 1$, if $|R_n| \leq \lambda t$ or n is larger than the maximal L value, where λ is a user-specified parameter for controlling the degree of data skew. For example, in Figure 4, supposing $t = 1$ and $\lambda = 1$, the iteration is stopped at step 2, in which there are two partitions remaining. After that, each final cluster will be generated by extending each start vertex in this cluster with all the vertices that have the best association with this start vertex. Finally, the function *DataDistributing* gives a novel strategy for selecting the partition to which the cluster is assigned. This strategy aims at a strong association and well-balanced distribution.

Association clustering generates clusters with different scales. The overhead of processing a small cluster is the same as that of a larger cluster. To avoid processing small clusters, we set a scale threshold for the clusters. A cluster that does not match this condition will be left behind in the sliding window and be processed when new data arrive. As each cluster may include one or more exemplars after association clustering, we want a unique exemplar to represent each cluster for data distributing. When one cluster has more than one exemplar, we select an exemplar from these exemplars through a secondary clustering: running AP again on these exemplars. If the result still has more than one exemplar, we choose the one with the maximum PageRank score. This can guarantee that the new generated exemplar has a relation with other vertices. Moreover, it prevents choosing a vertex with a large in-degree and results in skewed data placement.

Before the first bunch of data is processed, all storage nodes are empty. We use a greedy strategy for selecting the storage node to which cluster is assigned. This strategy aims at a well-balanced distribution. It sorts all clusters in decreasing order by the number of vertices they obtain. Then it assigns clusters one-by-one to minimal size partitions. It is notable that this strategy only processes little parts of the streaming graph and does not have effect over all distribution.

After initialization, the normal process of a cluster is to decide which storage node is to be distributed. First of all, we build an *ExIndex* for each storage node, which records each cluster's exemplar that has been already stored in the node in decreasing order by PageRank score, more specific information including the in-degree of the exemplar in order to approximate and update PageRank score. After association clustering, for each cluster, we send the PageRank of the exemplar PR_e to each storage node by multithread and compare the PageRank with the last one PR_i in the *ExIndex*. If $\text{PR}_e \geq \text{PR}_i$, we consider the cluster to be in association with data in this storage node and search the *ExIndex*. If this exemplar has already existed, we send a signal y back to the main node and otherwise n. According to all signals, the main node decides how to distribute the cluster. If more than one storage node sends y back, the cluster will be sent to the storage node with minimal storage capacity. If there is no signal y, then other clusters are processed first and the cluster is sent to the storage node with maximal storage space.

4. Experimental Evaluation

The experimental results are presented and analyzed in this section. Regarding the quality of partitions, we use *edge cut* (ec) size and *communication volume* (cv) with respect to optimizing graph query responding time. In addition, imbalance is also taken into consideration. We let imbalance be the fraction which equals the quotient between the size

Table 2: Statistics of real graphs.

Datasets	Amazon	EnWikt	DBLP	Yago
Vertices	735268	101355853	986285	2635316
Edges	5158014	4206756	6707236	5259414

Table 3: Edge cuts of real graphs.

Datasets	Amazon	EnWikt	DBLP	Yago
METIS	340163	30743970	1125114	1199683
HASH	5017956	95303805	6486516	4941314
DG	2254099	64543160	2918274	4674657
LDG	2204099	62750478	2663968	4239179
EDG	2252630	64543160	2958376	4614849
T	1292032	79535779	1292032	3388482
LT	1312032	79848943	1312032	3388662
ET	1310053	79624375	1310053	3388550
Assc	1113146	53410947	1154243	3254783

of the largest partition and the balanced (average) partition size.

We set random hash method as upper bound with the consideration that hash ignores both the structure of the graph and the locality among the edges completely. As a result, hash can achieve balanced partitions but incurs lots of edge-cutting across these partitions. We also set METIS as lower bound because it is an offline algorithm and can obtain the whole information of the graph to generate good partitions.

We also compare our solutions (short for Assc) to two typical kinds of partitioning algorithms: METIS (*gpmetis* with default configuration) as well as a collection of streaming partitioning algorithms including hash and methods proposed by Stanton and Kliot [14]. We also set the number of storage nodes k to a constant value equal to 16 and the value of parameter α to 0.5. We process unlabeled graphs with an edge weight as a constant value of 1.

All experiments were performed on a single machine which has a 16-core Intel Xeon CPU E7420 and 48 GB memory.

4.1. Real Graphs. We evaluate our approach over several real-world datasets which is generated from the real streaming case: *Amazon*, a symmetric graph describing the similarity among books; *English-Language Wiktionary*, a collaborative project to produce a free-content multilingual dictionary; *DBLP*, a bibliography service from which an undirected scientific collaboration network can be extracted; and *Yago*, a huge semantic knowledge base, derived from *Wikipedia*, *WordNet*, and *GeoNames*. Table 2 shows the basic information of the real graphs.

Tables 3 and 4 show the quality of the partitioning for the real-world graph. The quality is measured by ec and cv, respectively. The results were obtained by running the streaming partitioning methods and METIS. It is worth noting that we do not present communication volumes of other streaming graph partitioning methods in Table 4. As mentioned above, METIS is an offline algorithm that obtains the whole information of graph, whereas streaming graph partitioning methods can only pick up partial knowledge of the graph. It is reasonable that METIS surpasses streaming graph partitioning methods which can be proved by Table 3, so we omit cvs of other methods for succinctness.

Table 4: Communication volume situation of real graphs.

Datasets	Amazon	EnWikt	DBLP	Yago
METIS	407167	10021079	12227014	21026877
HASH	4316902	36827428	33183916	122343781
Assc	507354	14162380	15659341	15394621

Tables 3 and 4 give a close look at the edge-cutting number of each partitioning approach. It is clear that our solution produced partitions with significantly better quality reaching up to 1.17x (from 1.04x) over all other streaming partitioning methods, while METIS performs the best as we expected. The results in Table 3 also show that our method is comparable to METIS in terms of the number of edge cuts. We omit the experimental results of imbalance, for the reason that both our method and other competitive methods generated well-balanced partitions over all the graph datasets. It is worth noting that the maximal fraction represents imbalance equal to 0.04. So we ignore the results for briefness.

4.2. Synthetic Graphs. With the purpose of determining how effective our approach is in the context of graph query, we choose LUBM dataset with embedded communities to represent synthetic graphs. The LUBM dataset is widely used by the semantic web community for benchmarking triple stores. With the purpose of evaluating the scalability of our partitioning approach, we used six datasets of varying sizes generated by the LUBM data generator. Those LUBM datasets contain 10 M, 50 M, 100 M, 200 M, 300 M, and 500 M triples, respectively. We present the properties of these datasets in Table 5.

According to the experimental results, our solutions produced partitions with significantly better quality than HASH and METIS in terms of communication. Our method surpasses METIS at least 20% over all the LUBM datasets. For a straightforward view, we convert the data in these tables to Figure 5, which shows the communication traffic during the graph query with our system TripleBit [15, 16]. For the reason we explained above, we just present the results of METIS, HASH, and our methods.

Considering METIS holds the full knowledge of the graph, it is interesting that our method outperforms METIS just with fragmentary graph information acquired in streaming setting. It is worth noting that graph query is quite different from some typical graph algorithms like PageRank. It is the number of intrapartition edges and interpartition edges that dominates the execution time of PageRank. In this prerequisite, we can simply predict the performance of a graph algorithm with the quality of partitions split by one graph partitioner. However, graph query, unlike PageRank, is a subgraph patter matching problem. Its performance is more related to the distribution of subgraphs in each partition.

Table 5: Statistics of synthetic graphs.

Datasets	10 M	50 M	100 M	200 M	300 M	500 M
Vertices	3303724	16439317	32905170	65764621	98640459	164416780
Edges	13409395	66751196	133613894	267027610	400512826	667592614

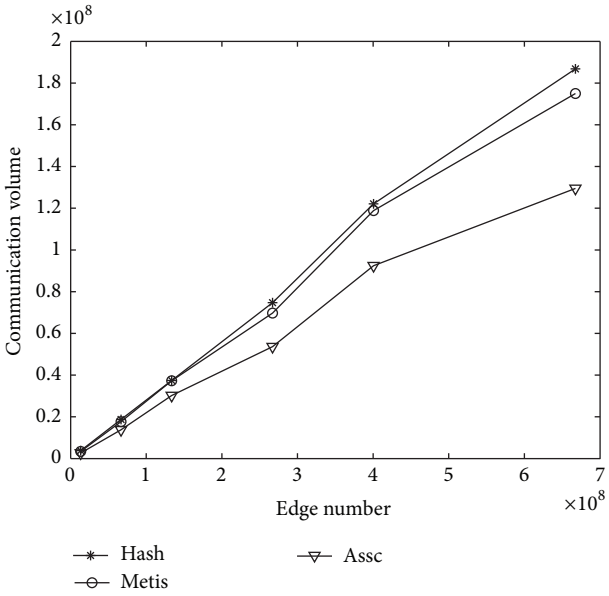


Figure 5: Communication during graph query.

Although METIS can lead to more balanced partitions with smaller fraction of edges cut, it ignores the subgraph associated with answers to graph query. Our method brings out larger number of edge cuts, admittedly. But by using *Association Based Clustering*, our method can preserve more subgraph pattern in each partition. In this way, when executing a graph query, a matched subgraph can be found within a partition boundary more likely. So compared to METIS, the communication traffic can be reduced by our method.

4.3. Discussion. METIS is based on a multilevel approach, which consists of three phases: the coarsening phase, the initial partitioning phase, and the uncoarsening phase. Among these three phases, coarsening phase plays a key role, which relies on the efficient choosing of objective functions, and for skewed graphs, many connected edges will be cut to avoid extremely large clusters. METIS can obtain the whole information of the graph structure and only the information of structure. As a result, it can generate good partitions with the cost of long execution time. As we explained above, in the scenario of graph query, it performs worse than our method.

Edge-based hash partitioning is a vertex-cut approach. This approach performs poorly when the graph structure is complex, which brings about the operations across partition boundaries. This consequently incurs frequent cross-node interactions coming with significant performance degeneration.

Our method considers the weight of edges by calculating PageRank score, which enables the vertices with high PageRank score to be merged late, giving a guarantee that we can get a reasonable size for the cluster. With association clustering, we can get high intra-associated clusters while the relations among vertices are well reserved, which means less *edge cut* and *communication volume*.

5. Related Work

We survey related work on static and streaming graph clustering from different angles. Regarding clustering problems, multiple algorithms have been proposed. K-means [17] is a simple and fast algorithm which often suffers from two major drawbacks. Firstly, the performance of K-means is highly related to the choice of the initial values which are treated as the cluster centers. Secondly, the objective function of the K-means is nonconvex coming with lots of local minima. K-means++ [18] algorithm was proposed which presents initialization procedure of cluster centers to overcome the initial centers sensitivity problem of a standard K-means. However, it also suffers from the local optimum problem. To add the ability for K-means++ to update its clusters as new points arrive from a data stream, StreamKM++ [19] was developed. StreamKM++ introduces the concept of a core set, and clusters are built on these core sets, which are good approximation of the clustering of the original points.

Being different from K-means, density-based clustering does not require the input of a predefined number of clusters and can form clusters with arbitrary shapes. One multiple stream density-based data processing algorithm is DenStream [20], which applies a damped window to DBSCAN [21]. DBSCAN is a popular density-based clustering algorithm with two simple parameters, which is effective, but is not designed to handle dynamic data streams. By extending the concept of core-objects to core-micro-clusters, DenStream [20] is able to cluster in the data stream environment.

Hash partitioning is one of the dominating approaches in graph partitioning. Lee and Liu present a novel semantic hash approach that utilizes access locality to partition big graphs across multiple computing nodes by maximizing the intrapartition processing capability and minimizing the interpartition communication cost [22]. Huang et al. use a graph partitioning algorithm instead of simple hash partitioning by source vertex, destination vertex, and labeled or unlabeled edge [23]. This allows vertices that are close together in the graph to be stored in the same machine. Stanton and Kliot proposed a weighted variant of the greedy algorithm for streaming balanced graph partitioning [14], which has an improvement over the hashing approach. Tsourakakis et al. presented a framework which unifies two seemingly orthogonal heuristics [24]. They also developed a streaming graph partitioning algorithm FENNEL based on this framework whose performance can be comparable to METIS.

Since each node in the graph has a natural PageRank property that can be used as a measure, regarding the computations performed over large graphs whose edges are presented in a streaming order, Sarma et al. proposed algorithms that require sublinear space and passes to compute the approximate PageRank values of vertices in a large graph [25]. Zhang et al. employed the affinity propagation algorithm and extend to data streaming to solve a challenge: how to cluster with the best representative [26]. Nguyen et al. present an algorithm that adopts eviction strategy to evaluate the likelihood of binding in terms of its contribution to a result in contrast to using a fixed time window for shedding the computation load [27]. Fischer et al. propose the use of graph partitioning algorithms to optimize the assignment of tasks to machines [28].

In order to process the growing data, many distributed infrastructures such as Hadoop [29] have been developed. Since Hadoop's batch-oriented synchronous nature does not satisfy the demand of real-time data processing, stream processing approaches based on information-flow processing have been proposed [28].

6. Conclusion and Future Work

In this paper, we propose a novel partitioning approach for streaming graph query in a general-purposed distributed storage system. Our approach, which is called association-oriented partitioning approach, adopts a hybrid approximate PageRank to retrace and compute associations between start vertices and other vertices. It then uses a hybrid PageRank-based affinity propagation clustering algorithm to generate several clusters, each including an exemplar. Finally, this method distributes different clusters with a brief strategy. This strategy judges whether the clusters have an association with the storage node. Extensive experiments conducted on these graphs prove that our method is effective. In the scenario of graph query, our method even outperforms METIS in terms of communication traffic across different partitions.

Our streaming graph partitioning approach needs to compute the rank values of vertices before distributing a vertex into a partition. The computation of PageRank value generally is a time-consuming step, although we have taken some strategies, including sliding window and approximate PageRank computation to improve the performance. Thus, our work will continue in two directions. *First*, we are working on extending our approach to distributed computing architecture. *Second*, we are interested in exploring the querying feature for labeled streaming graphs based our system.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The paper was recommended to be published in journals by International Workshop on Big Data Programming and System Software 2016 (BDPSS 2016). The research is supported by Science and Technology Planning Project of Guangdong Province, China (nos. 2016B030306003 and 2016B030305002).

References

[1] "Linked open data project", 2014, http://linkeddata.org/.

[2] P. Wang, B. Xu, Y. Wu, and X. Zhou, "Link prediction in social networks: the state-of-the-art," *Science China Information Sciences*, vol. 58, no. 1, pp. 1–38, 2014.

[3] Z. Kaoudi, K. Kyzirakos, and M. Koubarakis, "Sparql query optimization on top of dhts," in *Proceedings of 9th International Semantic Web Conference (ISWC '10)*, pp. 418–435, Springer, 2010.

[4] A. Harth, J. Umbrich, A. Hogan, and S. Decker, "YARS2: a federated repository for querying graph structured data from the Web," *The Semantic Web*, vol. 4825, pp. 211–224, 2007.

[5] P. Yuan, W. Zhang, C. Xie, H. Jin, L. Liu, and K. Lee, "Fast iterative graph computation: a path centric approach," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, (SC '14)*, pp. 401–412, USA, November 2014.

[6] P. Yuan, C. Xie, L. Liu, and H. Jin, "Pathgraph: a path centric graph processing system," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 10, pp. 2998–3012, 2016.

[7] G. Malewicz, M. H. Austern, A. J. C. Bik et al., "Pregel: a system for large-scale graph processing," in *Proceedings of the International Conference on Management of Data (SIGMOD '10)*, pp. 135–146, ACM, June 2010.

[8] B. Wu, Y. Zhou, P. Yuan, L. Liu, and H. Jin, "Scalable SPARQL querying using path partitioning," in *Proceedings of the IEEE 31st International Conference on Data Engineering (ICDE '15)*, pp. 795–806, IEEE Computer Society, Seoul, South Korea, April 2015.

[9] "Metis", 2015, http://www.cs.umn.edu/metis/.

[10] "Lubm", 2014, http://swat.cse.lehigh.edu/projects/lubm/.

[11] S. Fortunato, M. Boguñá, A. Flammini, and F. Menczer, "Approximating pagerank from in-degree," in *Proceedings of 4th International Workshop on Algorithms and Models for the Web-Graph (WAW '06)*, vol. 4936, pp. 59–71, Springer, 2006.

[12] Y. Guo, Z. Pan, and J. Heflin, "LUBM: a benchmark for OWL knowledge base systems," *Web Semantics*, vol. 3, no. 2-3, pp. 158–182, 2005.

[13] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *American Association for the Advancement of Science. Science*, vol. 315, no. 5814, pp. 972–976, 2007.

[14] I. Stanton and G. Kliot, "Streaming graph partitioning for large distributed graphs," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2012*, pp. 1222–1230, chn, August 2012.

[15] P. Yuan, P. Liu, B. Wu, H. Jin, W. Zhang, and L. Liu, "Triplebit: a fast and compact system for large scale rdf data," *Proceedings of the VLDB Endowment*, vol. 6, no. 7, pp. 517–528, 2013.

[16] P. Yuan, C. Xie, H. Jin, L. Liu, G. Yang, and X. Shi, "Dynamic and fast processing of queries on large-scale rdf data," *Knowledge and Information Systems (KAIS)*, vol. 41, no. 2, pp. 311–334, 2014.

[17] G. Krishnasamy, A. J. Kulkarni, and R. Paramesran, "A hybrid approach for data clustering based on modified cohort intelligence and K-means," *Expert Systems with Applications*, vol. 41, no. 13, pp. 6009–6016, 2014.

[18] D. Arthur and S. Vassilvitskii, "k-means++: the advantages of careful seeding," in *Proceedings of the 18th Annual ACM-SIAM*

Symposium on Discrete Algorithms (SODA '07), pp. 1027–1035, Society for Industrial and Applied Mathematics (SIAM), 2007.

[19] M. R. Ackermann, Märtens M., C. Raupach, K. Swierkot, C. Lammersen, and C. Sohler, "StreamKM++: a clustering algorithm for data streams," *ACM Journal of Experimental Algorithmics*, vol. 17, Article 2.4, 30 pages, 2012.

[20] F. Cao, M. Ester, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in *Proceedings of the 6th SIAM International Conference on Data Mining (SDM '06)*, pp. 328–339, SIAM, 2006.

[21] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the 2th International Conference on Knowledge Discovery and Data Mining (KDD '96)*, pp. 226–231, 1996.

[22] K. Lee and L. Liu, "Scaling queries over big rdf graphs with semantic hash partitioning," *Proceedings of the VLDB Endowment*, vol. 6, no. 14, pp. 1894–1905, 2013.

[23] J. Huang, D. J. Abadi, and K. Ren, "Scalable SPARQL querying of large RDF graphs," *Proceedings of the VLDB Endowment*, vol. 4, no. 11, pp. 1123–1134, 2011.

[24] C. Tsourakakis, C. Gkantsidis, B. Radunovic, and M. Vojnovic, "FENNEL: streaming graph partitioning for massive scale graphs," in *Prcedings of the 7th ACM International Conference on Web Search and Data Mining, WSDM 2014*, pp. 333–342, USA, February 2014.

[25] A. D. Sarma, S. Gollapudi, and R. Panigrahy, "Estimating PageRank on graph streams," *Journal of the ACM*, vol. 58, no. 3, Article ID 13, 2011.

[26] X. Zhang, C. Furtlehner, C. Germain-Renaud, and M. Sebag, "Data stream clustering with affinity propagation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 7, pp. 1644–1656, 2014.

[27] M. K. Nguyen, T. Scharrenbach, and Bernstein A., "Seven commandments for benchmarking semantic flow processing systems," in *Proceedings of the 9th International Conference on Scalable Semantic Web Knowledge Base Systems (SSWS '13)*, pp. 66–80, Springer, 2013.

[28] L. Fischer, T. Scharrenbach, and A. Bernstein, "Scalable linked data stream processing via network-aware workload scheduling," in *Proceedings of the 9th International Conference on Scalable Semantic Web Knowledge Base Systems (SSWS '13)*, pp. 81–96, 2013.

[29] J. Shafer, S. Rixner, and A. L. Cox, "The hadoop distributed filesystem: balancing portability and performance," in *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS '10)*, pp. 122–133, usa, March 2010.

Hindawi
Scientific Programming
Volume 2017, Article ID 3787053, 15 pages
https://doi.org/10.1155/2017/3787053

Research Article

Clustering Classes in Packages for Program Comprehension

Xiaobing Sun,[1,2] Xiangyue Liu,[1] Bin Li,[1] Bixin Li,[3] David Lo,[4] and Lingzhi Liao[5]

[1]*School of Information Engineering, Yangzhou University, Yangzhou, China*
[2]*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China*
[3]*School of Computer Science and Engineering, Southeast University, Nanjing, China*
[4]*School of Information Systems, Singapore Management University, Singapore*
[5]*Nanjing University of Information Science & Technology, Nanjing, China*

Correspondence should be addressed to Bin Li; lb@yzu.edu.cn

Received 16 October 2016; Revised 13 February 2017; Accepted 27 February 2017; Published 11 April 2017

Academic Editor: Xuanhua Shi

During software maintenance and evolution, one of the important tasks faced by developers is to understand a system quickly and accurately. With the increasing size and complexity of an evolving system, program comprehension becomes an increasingly difficult activity. Given a target system for comprehension, developers may first focus on the package comprehension. The packages in the system are of different sizes. For small-sized packages in the system, developers can easily comprehend them. However, for large-sized packages, they are difficult to understand. In this article, we focus on understanding these large-sized packages and propose a novel program comprehension approach for large-sized packages, which utilizes the Latent Dirichlet Allocation (LDA) model to cluster large-sized packages. Thus, these large-sized packages are separated as small-sized clusters, which are easier for developers to comprehend. Empirical studies on four real-world software projects demonstrate the effectiveness of our approach. The results show that the effectiveness of our approach is better than Latent Semantic Indexing- (LSI-) and Probabilistic Latent Semantic Analysis- (PLSA-) based clustering approaches. In addition, we find that the topic that labels each cluster is useful for program comprehension.

1. Introduction

Program comprehension is one of the most frequently performed activities in software maintenance [1, 2]. It is a process whereby a software practitioner understands a program using both knowledge of the domain and semantic and syntax knowledge, to build a mental model of the program [3, 4]. Developers working on software maintenance tasks spend around 60% of their time for program comprehension [5]. As software evolves, its complexity becomes increasingly higher. Moreover, some documents affiliated to the system also become outdated or inaccessible, which makes program comprehension more difficult.

In practice, the natural top-down program comprehension process can effectively facilitate developers to understand the system step by step [6]. For an object oriented Java software system, developers can also understand a system in such a top-down way. Packages are first taken into consideration. Then, interesting packages are selected, and developers further go deep into classes in the packages. For small-sized packages (with several classes), it is easy for developers to understand them. However, for packages with many classes in them, it is more challenging for developers to understand these classes, their relationships, and their functionalities [7, 8]. To aid program understanding, classes in these large-sized packages can be clustered into smaller-sized groups. With such clustering, developers can understand a system more easily.

There are several approaches that cluster programs based on static structural dependencies in the source code [9]. Static structural dependencies based approaches usually cluster classes in a system based on static structural dependencies among program elements, such as variable and class references, procedure calls, use of packages, and association and inheritance relationships among classes [8, 10, 11]. These approaches are more suitable in the process of implementing a change request in the source code. But before implementing a change request in the source code, developers should know

which part in the source code is related to the change request. Specifically, they need to know the functional points of a system and where in the source code corresponds to these functional features. A feature or functional point represents a functionality that is defined by requirements and accessible to developers and users. Then, they can implement source code level changes. Hence, some studies focused on understanding the functional features of a system and proposed semantic clustering, which exploits linguistic information in the source code, such as identifier names and comments [12]. These approaches usually take the whole system as input and generate clusters at some granularity levels, for example, class level or method level. The generated clusters corresponding to different functional features are used to divide a system into different units [13, 14]. This article also focuses on exploiting linguistic information in the source code to understand functional features of different clusters in large-sized packages. In a large-sized package, there are a number of functional features or concerns. Each of these concerns is implemented in a set of classes. The previous studies focused on clustering a software unit. However, developers still do not easily know what the functional features that each cluster expresses are. So to get a good understanding of its concerns and the classes that implement each of them, in this article, we further generate a set of topics to describe each cluster.

This article proposes a technique to generate a set of clusters of classes for a large-sized package, where different clusters correspond to different functional features or concerns. Our approach is based on Latent Dirichlet Allocation (LDA), which is a topic model and one of the popular ways to analyze unstructured text in the corpus [15]. LDA can discover a set of ideas or themes that well describe the entire corpus. We use LDA for a whole package and extract latent topics to capture its functional features. Then, classes in the package with similar topics are clustered together.

Our approach can be effectively used for top-down program comprehension during software maintenance. For small-sized packages, developers can directly understand them. For large-sized packages, our approach can be used to divide packages into small clusters. Each of these small clusters can be understood more easily than the original large-sized package. The main contributions of this article are as follows:

(1) We propose to use LDA to generate clusters for large-sized packages. The topics generated by LDA are useful to indicate the functional features for these class clusters.

(2) We conduct an empirical study to show the effectiveness of our approach on four real-world open-source projects, *JHotDraw*, *jEdit*, *JFreeChart,* and *muCommander*. The results show that our approach is more effective in identifying more relevant classes in the cluster than other semantic clustering approaches, that is, Latent Semantic Indexing- (LSI-) and Probabilistic Latent Semantic Analysis- (PLSA-) based clustering.

(3) The empirical study on four selected packages from four subject systems shows that the topics generated by our approach are useful to help developers understand these packages.

The rest of the article is organized as follows: in the next section, we introduce the background of program comprehension and LDA model. Section 3 describes our approach. We describe the design of our experiment, experiment results, and threats to validity of our study in Sections 4, 5, and 6, respectively. In Section 7, related work using clustering for program comprehension is discussed. Finally, we conclude the article and outline directions for future work in Section 8.

2. Background

In this article, we use LDA to cluster classes in large-sized packages for easier program comprehension. This section discusses the background of program comprehension and LDA topic model.

2.1. Program Comprehension. For software developers, program comprehension is a process whereby they understand a software artifact using both knowledge of the domain and semantic and syntax knowledge [10]. Program comprehension can be divided into bottom-up comprehension, top-down comprehension, and various combinations of these two processes. In bottom-up comprehension, a developer may first read all the source code at finer statement or method level and abstract features and concepts according to the low-level information. Then, coarser class-level or package-level elements are read and understood. Finally, developers comprehend the whole system. For top-down comprehension, a developer first utilizes knowledge about the domain to build a set of expectations that are mapped to the source code. Then, he/she understands the coarser package-level or class-level elements, followed by finer method-level or statement-level elements. Finally, the developer also gets an understanding of the whole system. In practice, top-down program comprehension is more acceptable since it meets humans' way of thinking from simple to complex, from whole to part [3].

Software clustering is one of the effective techniques for top-down program comprehension. During software maintenance, developers usually need to identify the functional features they are interested in to help them accomplish a change request. In this article, we propose a software clustering technique using LDA to provide some features for developers to facilitate the top-down program comprehension process.

2.2. Latent Dirichlet Allocation. Topic models were originated from the field of information retrieval (IR) to index, search, and cluster a large amount of unstructured and unlabeled documents. A topic is a collection of terms that cooccur frequently in the documents of the corpus. One of the mostly used topic models in software engineering community is Latent Dirichlet Allocation (LDA) [16–18]. It requires no training data and can well scale to thousands or millions of documents.

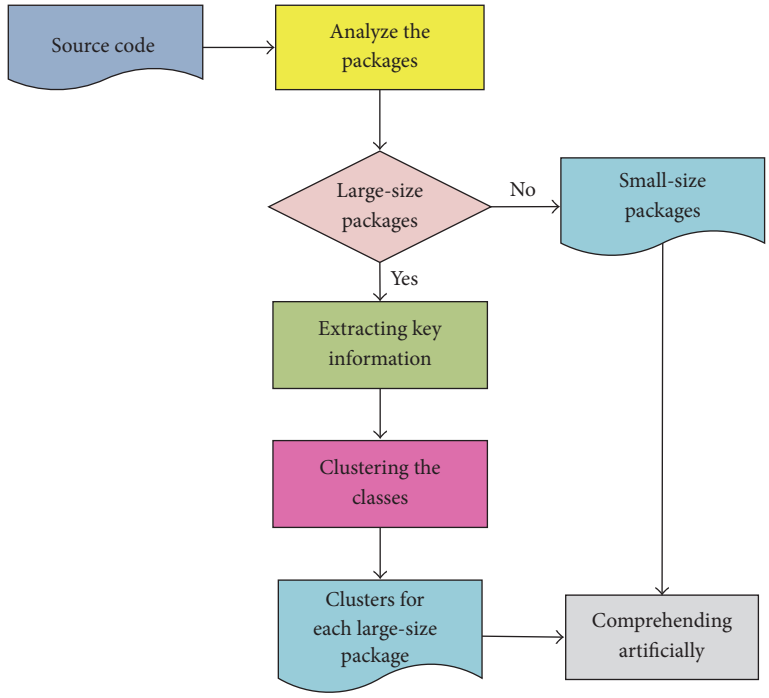


Figure 1: Process of our approach.

LDA models each document as a mixture of K corpus-wide topics and each topic as a mixture of terms in the corpus [15]. More specifically, it means that there is a set of topics to describe the entire corpus; each document can contain more than one of these topics; and each term in the entire repository can be contained in more than one of these topics. Hence, LDA is able to discover a set of ideas or themes that well describe the entire corpus. It assumes that documents have been generated using the probability distribution of the topics and that words in the documents were generated probabilistically in a similar way.

In order to apply LDA to the source code, we represent a software system as a collection of documents (i.e., classes) where each document is associated with a set of concepts (i.e., topics). Specifically, the LDA model consists of the following building blocks:

(1) A word is the basic unit of discrete data, defined to be an item from a software vocabulary $V = \{w_1, w_2, \ldots, w_v\}$, such as an identifier or a word from a comment.

(2) A document, which corresponds to a class, is a sequence of words denoted by $d = \{w_1, w_2, \ldots, w_n\}$, where w_i is the ith word in the sequence.

(3) A corpus is a collection of documents (classes) denoted by $D = \{d_1, d_2, \ldots, d_m\}$.

Given m documents containing k topics expressed over v unique words, the distribution of ith topic t_i over v words and the distribution of jth document over k topics can be represented.

By using LDA, it is possible to formulate the problem of discovering a set of topics describing a set of source code classes by viewing these classes as mixtures of probabilistic topics. For further details on LDA, interested readers are referred to the original work of Blei et al. [15].

With LDA, latent topics can be mined, allowing us to cluster them on the basis of their shared topics. In this article, to effectively use LDA, we apply it in a package-level corpus rather than each class to extract the latent topics to simulate the functional features or concerns for a package since small (class-level) corpus is too small to generate good topics [19–23]. Then, we cluster the classes according to these topics and assign different classes to their corresponding topics [23].

3. Our Approach

Faced with the source code of a software system, developers need to use their domain knowledge to comprehend the program from coarse package level to class level in each package. The process of understanding different packages is different. In the process, small-sized packages are easy to understand while large-sized packages are complex and they need to be separated into small-sized clusters. In this article, we focus on clustering the classes in large-sized packages as well as their corresponding functional features.

The process of understanding packages is described in Figure 1. Firstly, we analyze the size of each package in the software system. Small-sized packages are comprehended manually. For large-sized packages, there are two steps. First, LDA is used to extract the latent key information to facilitate the comprehension process. Then, on the basis of the key information of each package, we adopt the clustering to build small-sized clusters to decompose each package. Thus, given the source code of a software system at hand, programmers

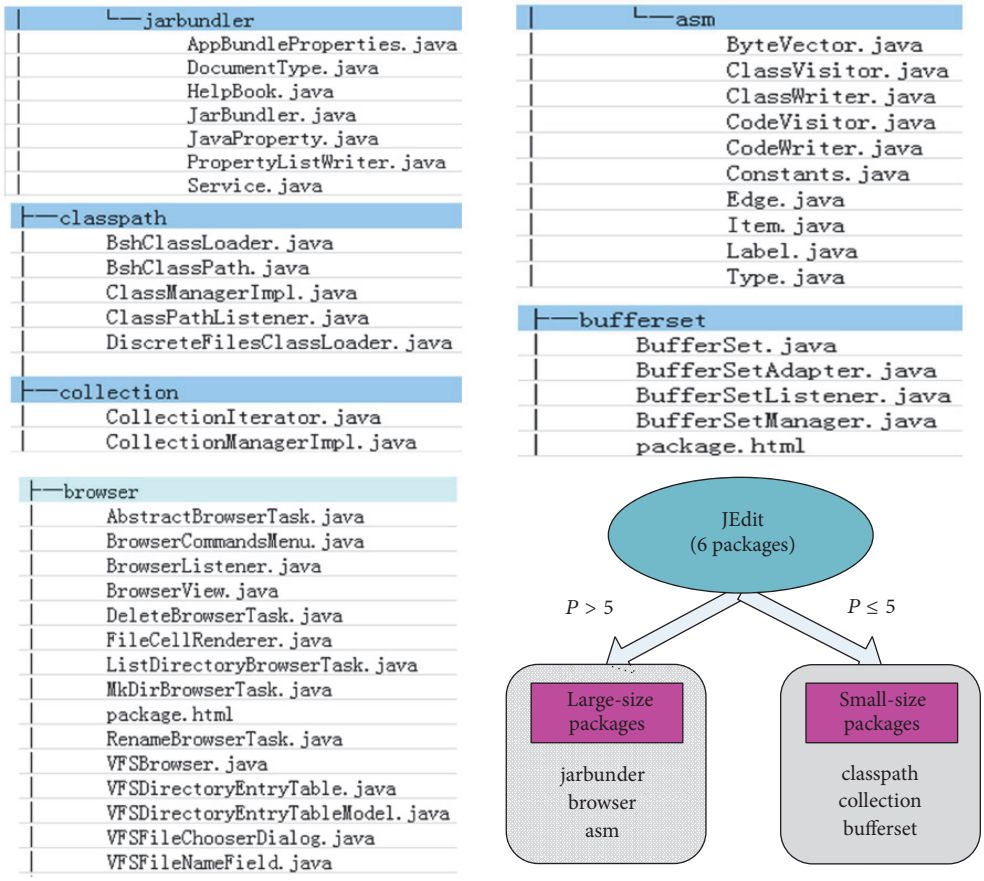


Figure 2: An example of separating packages into large-sized packages and small-sized packages for six packages in jEdit when P is set to 5.

can comprehend small-sized packages by themselves and large-sized packages with the help of our approach. In the following subsections, we discuss more details of our approach.

3.1. Analyzing the Size of Packages. Our approach focuses on understanding large-sized packages. So we first need to select large-sized packages in a program. Here, we set a threshold P for the number of classes in a package. The packages including more than P classes are selected for analysis. These packages are separated into smaller clusters to facilitate program comprehension. Figure 2 shows an example of separating packages into large-sized packages and small-sized packages for six packages in *jEdit* when P is set to 5. The packages *jarbunder*, *browser,* and *asm* are classified as large-sized packages.

3.2. Extracting Key Information Based on LDA. During the program comprehension process, developers are more focused on functional features or concerns of the program. In the program, the source code contains not only the syntax information but also the unstructured data, such as natural language identifiers and comments [24]. These unstructured source code identifiers and comments can be used to capture the semantics of the developers' intent [25]. They represent an important source of domain information and can often serve as a starting point in many program comprehension tasks [26, 27]. However, there exists noise in the source code, which can potentially confuse the LDA application. So natural language processing (NLP) techniques are usually used to perform one or more preprocessing operations before applying LDA models to the source code data. Then, LDA can be effectively used to generate the topics. To effectively use LDA, we apply it in a package-level corpus to simulate the functional features or concerns for a package. Finally, we cluster the classes according to these topics and assign different classes to their corresponding topics.

3.2.1. Preprocessing of the Source Code. There are several typical preprocessing operations for the unstructured part of a source code. These preprocessing operations can be performed to reduce noise and improve the quality of the resulting text for LDA [28].

We first isolate identifiers and comments and strip away syntax and programming language keywords (e.g., "*public*" and "*int*"). First, we remove header comments since they often include generic information about the software that are included in most of source code files. Then, we tokenize each word based on common naming practice, such as camel

Figure 3: The process of preprocessing the class *InvalidHeaderException.java* in jEdit.

case *("oneTwo")* and underscores *("one_two")*, and remove common English language stop words *(the*, *it*, and *on)* and programming language related key words *(public*, *int*, and *while)*.

After preprocessing the unstructured part of source code files, LDA can be used to extract key information more effectively. Figure 3 shows an example of the detailed process of preprocessing the source code in the class *InvalidHeaderException.java* in *jEdit*. After preprocessing the source code, most of the useful words are kept for LDA application.

3.2.2. Extracting Key Information from Large-Sized Packages. After preprocessing each class in large-sized packages, we need to extract key information from them. LDA is an effective approach to discover a set of ideas or themes that well describe the entire corpus. Before using LDA, we need to set the number of topics, that is, K. This parameter affects the effectiveness of LDA application. In this article, K is related to the size of clusters for a package, which is determined by users.

An LDA application generates two files: one is the word-topic matrix which lists the words for each topic and the other is the topic-document matrix which shows the percentage of topics in each document, also called the membership value. An example is given in Figure 4. The results show the distribution of different topics in different classes, and each topic is described by different words with different possibilities. These topics express different functional features of classes in the package.

3.3. Clustering Classes in Large-Sized Packages. After extracting the topics from the objective package, classes having similar topics should be allocated in a cluster to aid their comprehension. In this subsection, we discuss details for clustering classes in a package.

3.3.1. Generating Initial Clusters. To cluster classes in a package, the number of clusters should be first determined. However, it is difficult to know this information at the

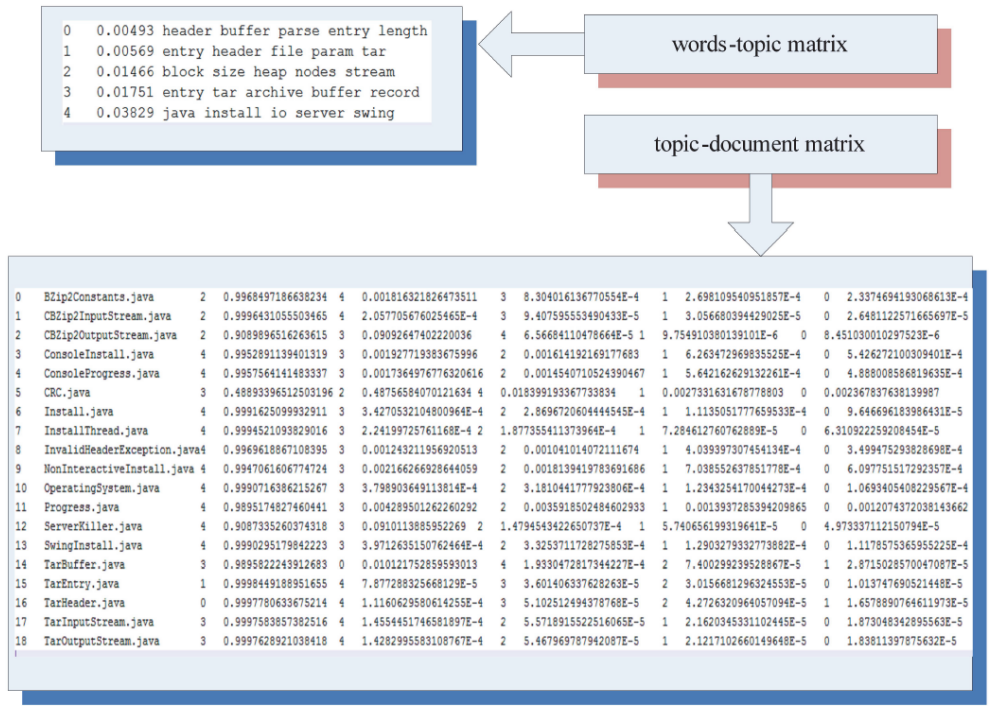


Figure 4: An example of the output of an LDA application.

beginning. In this article, the number of clusters is estimated based on the number of classes in a cluster.

Let us assume that the number of classes in an initial cluster is M, where M is a user-defined parameter. That is, if a user thinks that an M-scale cluster is easy for him/her to understand, he/she can set the initial size of each cluster as M, for example, 5 and 10. For a package with N classes ($N \geq M$), each of these classes should be put into a cluster. Thus there will be $\lceil N/M \rceil$ (a whole number) clusters for a package. We set the number of topics K to be the same as the number of clusters (i.e., $\lceil N/M \rceil$), because we need a topic to label each cluster.

After applying the LDA in a preprocessed package, we get two files, the word-topic matrix which lists the words for each topic and the topic-document matrix which shows the percentage of the topic words in each document. To allocate different classes to their corresponding topics, we use the topic-document matrix. That is, we allocate the top M documents to these K topics in the topic-document matrix. Thus a set of clusters can be generated, which we call the *initial clusters* for a package.

The ideal situation for the initial clusters is that each class is just assigned to only its own and exclusive cluster. Inevitably, there are two special cases; one is that a class may match different topics in the topic-document matrix. Such classes are called shared classes, which we need to reassign. The other case is that there may be some remaining classes that are not included in the top M documents in any topics. Such classes are called nonmatching classes, which need to be assigned to the most probable clusters related to them. In the following, we deal with these classes to guarantee that each class is assigned to one and only one cluster.

3.3.2. Assigning Shared Classes and Nonmatching Classes. Shared classes are the classes matching different topics in the generated topic-document matrix. These classes are all listed in the top M classes for each topic. We list all the classes shared by different topics and the membership value of each topic for them. A shared class is allocated to the cluster corresponding to the topic with the highest membership value.

For nonmatching classes that are not initially matched to any cluster, they are processed in a similar way as shared classes. We list all these nonmatching classes and their membership values. Each of these nonmatching classes is put into the cluster corresponding to the topic with the highest membership value.

Finally, each cluster in a package only contains classes having high membership values and each class is a member of only one cluster. Based on the word-topic matrix, we can see the words describing the topic, which indicates the feature of the cluster. Figure 5 shows an example of the process to generate the clusters for a large-sized package. First, initial clusters are generated according to the membership values with five topics. Then, shared classes and nonmatching classes are assigned to corresponding initial clusters based on their membership values. Finally, a set of clusters for the large-sized package is obtained as shown in the bottom-left part of Figure 5.

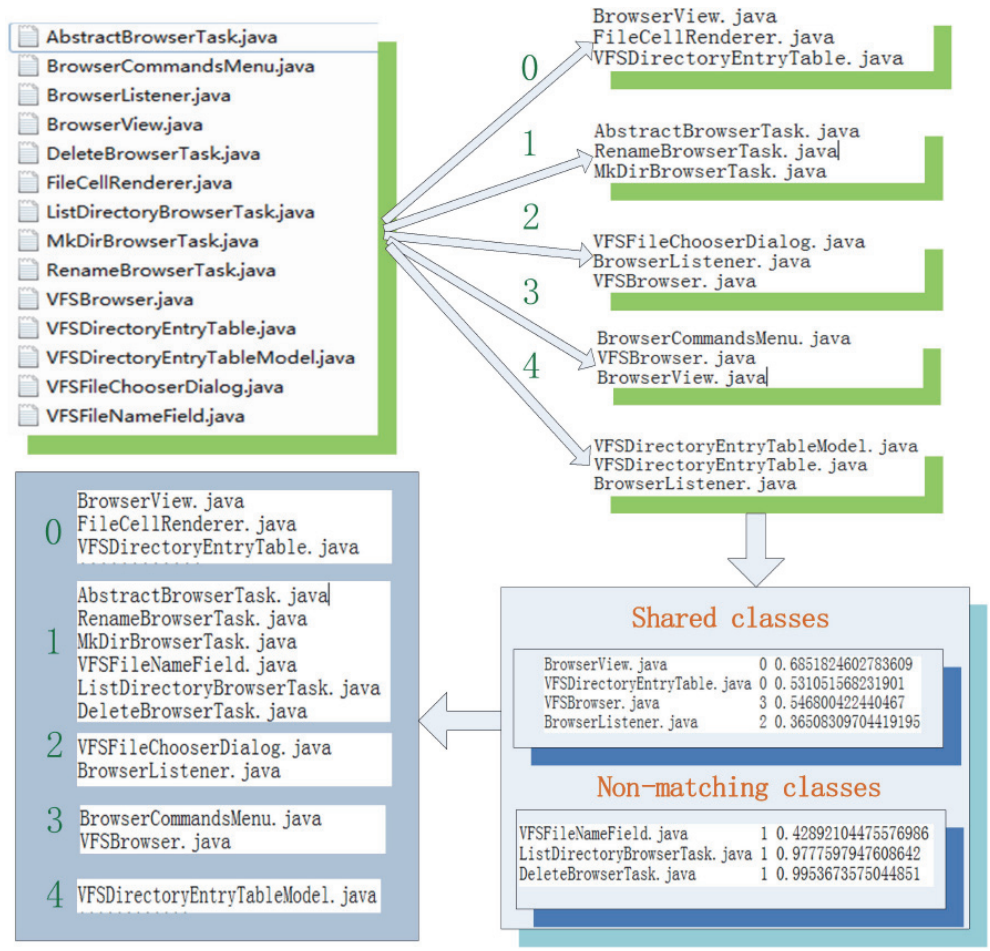


Figure 5: An example of generating clusters for a large-sized package (with the number of topics set to 5).

4. Case Study

In this section, we conduct case studies to evaluate the effectiveness of our approach. In our study, we address the following three research questions *(RQs)*:

> *RQ1*: Does the number of topics affect the shared classes and nonmatching classes?
>
> *RQ2*: Is our LDA clustering approach more effective than other semantic clustering approaches, that is, LSI-based clustering and PLSA-based clustering?
>
> *RQ3*: Can our approach provide useful topics for developers to understand the classes in the package(s)?

In our approach, the number of topics is set by users themselves. We investigate *RQ1* to see how this parameter affects the number of shared and nonmatching classes. In addition, we investigate *RQ2* to see whether our clustering approach using LDA is more effective than other semantic clustering based approaches based on LSI and PLSA [12, 29–31], respectively. Finally, there is another difference between our approach and other clustering approaches; that is, each cluster that is generated by our approach is labeled with a topic to facilitate understanding of the cluster. So *RQ3* aims to answer whether the topic labeling each cluster can help developers understand the cluster or not.

4.1. Empirical Environment. We implemented our approach with Java language in the Eclipse environment. In addition, all the selected subject programs are also Java programs. So our case study was conducted in the Eclipse environment.

4.2. Subject Systems. We address our research questions by performing case studies on the source code of four well-known software systems, *JHotDraw* (https://sourceforge.net/projects/jhotdraw), *jEdit* (https://sourceforge.net/projects/jedit), *JFreeChart* (https://sourceforge.net/projects/jfreechart), and *muCommander* (http://www.mucommander.com), as shown in Table 1.

JHotDraw is a medium-sized, open-source, 2D drawing framework developed in the Java programming language. *jEdit* is a medium-sized, open-source text editor written in Java. *JFreeChart* is a free 100% Java chart library that makes it easy for developers to display professional quality charts

Table 1: Subject systems.

Subject	Version	Files	Packages	Classes
JHotDraw	7.0.6	144	23	305
jEdit	5.1.0	147	43	573
JFreeChart	1.0.17	105	70	990
muCommander	0.9.0	98	89	692

Table 2: The percentage of classes over P (5, 10, and 15) of the four systems.

Subject	$P > 5$ (%)	$P > 10$ (%)	$P > 15$ (%)
JHotDraw	91.8	79.7	68.5
jEdit	91.1	81.3	72.2
JFreeChart	93.2	81.0	67.6
muCommander	81.1	56.9	37.2
Average	89.3	74.7	61.4

in their applications. *muCommander* is a lightweight, cross-platform file manager that runs on any operating system supporting Java.

These projects belong to different problem domains. They are general enough to represent real-world software systems, and they have been widely used in empirical studies in the context of software maintenance and evolution [32, 33]. In addition, they have become the de facto standard system for experiments and analysis in topic and concern mining (e.g., by Robillard and Murphy [34] and Binkley et al. [35]). Moreover, these four subject systems of different sizes that are neither too small nor too large are selected due to their good design and manageable size for manual analysis.

4.3. Parameters Setting. In our approach, there are two parameters, P and K. P represents the size of a package and K is the number of topics as input for the LDA model. Values of these two parameters will affect the number of packages to be subdivided into clusters and the number of clusters in a package. Table 2 shows the percentage of classes in packages with different number of classes. From the results, when P is 5, 10, and 15, the average percentages of classes in packages of large sizes are 89.3%, 74.7%, and 61.4%, respectively. In this study, we consider packages of size larger than 10 as the large-sized packages used to evaluate our approach. Hence, for all four systems, most of classes and packages are used to evaluate our approach.

The other parameter in our approach is the number of topics (K) for LDA analysis. K is an important parameter which also indicates the number of clusters for the final clustering results. It determines the size of each cluster. We set K to be 5, 10, and 15 for our study, respectively.

4.4. Methods and Measures. For LDA computation, we used *MALLET* (http://mallet.cs.umass.edu), which is a highly scalable Java implementation of the *Gibbs* sampling algorithm. We ran for 10,000 sampling iterations, the first 1000 of which were used for parameter optimization. We selected different numbers of topics to use *MALLET* to generate the word-topic matrix and topic-document matrix. Then, we clustered each large-sized package based on these two matrixes.

For semantic clustering based on LSI and PLSA [12, 29–31] that we used to compare with our approach, they are popular methods for cluster analysis, especially for clustering nonstructured data. LSI uses singular value decomposition to explore patterns in the relationships between the terms and concepts contained in an unstructured corpus [36]. LSI is implemented based on the assumption that words used in the same contexts tend to have similar meanings. Hence, LSI is able to extract the conceptual contents from a corpus by establishing associations between those terms that occur in similar contexts. Probabilistic Latent Semantic Analysis (PLSA) is a statistical technique for the data analysis, which is based on a mixture decomposition derived from a latent class model [30, 31].

We selected these clustering approaches for comparison because (1) they are widely used in clustering software data and show promising results [37, 38] and (2) they are also clustering approaches based on lexical information which is similar to our approach. In our study, they are performed by clustering classes with similar vocabularies. After calculating the similarity between each pair of documents, an agglomerative hierarchical clustering algorithm is executed. There are a lot of similarity measures, for example, cosine similarity, Manhattan distance, and Euclidean distance [39]. Cosine similarity which is a popular similarity measure is used here [33, 40].

To answer *RQ1*, we compute the number of shared classes and nonmatching classes and the *shared occurrence counts* (or *shared counts*) of the shared classes. For example, if one class is shared by topic 1 and topic 2, its shared count is 1. If the class is also shared by topic 3, the shared count is 2. We analyze the percentages of shared classes and nonmatching classes as well as the shared counts for different numbers of topics.

For *RQ2*, our study involved 10 participants from university and industry. Half of them are from our lab with 2-3 years of development experience and the other half are from industry with 5-6 years of development experience especially

Table 3: The selected packages and selected classes.

Subject	Package	Size	Class	K
JHotDraw	*JHotDraw.src.org.jhotdraw.app.action*	33	*AbstractProjectAction.java*	10
jEdit	*jEdit.org.gjt.sp.jedit.gui*	73	*AbbrevEditor.java*	15
JFreeChart	*jfreechart.source.org.jfree.chart.plot*	52	*AbstractPieLabelDistributor.java*	10
muCommander	*muCommander.main.com.mucommander.command*	15	*AssociationBuilder.java*	5

large project development experience. They are not familiar with the systems before. Then, they were assigned with a class as shown in the fourth column of Table 3. The task for them is to identify the most likely classes that are related to the given classes in their enclosing packages. Then each participant obtained a cluster of classes for each given class. As different participants may generate different clusters, they needed to discuss the results and reached a consensus on the clustering results for each given class. We used the clustering results as the authoritative clusters to compare with the clusters produced by our approach and the LSI-based/PLSA-based clustering approach. For LSI-based clustering approach and our approach that are used for comparison, we need to set the K value. Based on the size of the authoritative clusters, we set the K values for the packages, which are shown in the last column of Table 3. To answer *RQ2*, we first provided the clustering results of the three approaches to participants. In this process, to guarantee a fair treatment, they did not know which clustering results were generated by our approach or the LSI-based/PLSA-based clustering approach. Then, each of participants assessed each of the three clusters to vote the best one. In addition, to quantitatively compare these two approaches, we used precision and recall, two widely used information retrieval and classification metrics [41], to validate the accuracy of different clustering approaches. Precision measures the fraction of classes identified by a clustering approach to be in the same cluster as the given class that are truly relevant (based on the authoritative cluster), while recall measures the fraction of relevant results (i.e., classes that appear in the authoritative cluster) that are put in the same cluster as the given class by a clustering approach. Mathematically, they are defined as follows:

$$\begin{aligned} &\text{Precision} \\ &= \frac{\left|\text{Clustering results} \cap \text{Authoritative results}\right|}{\left|\text{Clustering results}\right|} \\ &\times 100\% \\ &\text{Recall} \\ &= \frac{\left|\text{Clustering results} \cap \text{Authoritative results}\right|}{\left|\text{Authoritative results}\right|} \\ &\times 100\%. \end{aligned} \tag{1}$$

In the above equations, clustering results and authoritative results are sets of classes.

To answer *RQ3*, participants were required to write the words in the identifiers or comments to label the authoritative clusters. This process is similar to that of *RQ2*, and a set of authoritative words are produced. To show whether the topics generated by our approach are useful, the participants needed to assess the generated topics to check whether they are useful for them to understand the clusters. Each participant needs to provide a rating in a five-point Likert scale, 1 (very useless) to 5 (very useful). Finally, we also computed the precision and recall of the words in the topics by comparing them with authoritative words. The way precision and recall are computed is similar to the way they are computed to answer *RQ2*.

Overall, the participants needed to answer four questions during the evaluation process. In *RQ2*, they were asked to give the answers of the authoritative results for the clustering and assess the results between *LSI* or *PLSI* and our approach. In *RQ3*, they needed to provide labels of the clusters and assess the topics generated by our approach.

5. Results

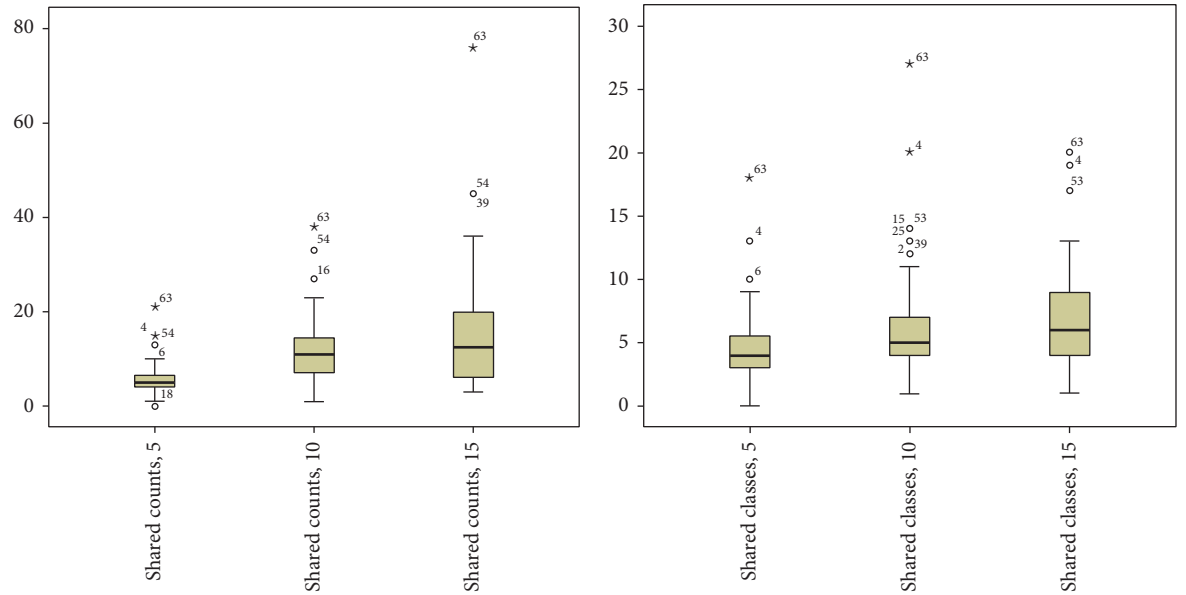
In this section, we gather and analyze results collected from the case studies to answer *RQ1*, *RQ2,* and *RQ3*.

5.1. RQ1. First, we see the existence of shared classes and nonmatching classes in the initial clusters. Table 4 shows the average percentage of the initial clusters without nonmatching classes and shared classes. From the results, we see that there do exist some shared classes and nonmatching classes in the initial clusters. So we need to perform the operation of reassigning these shared classes and nonmatching classes. Then, we see how the number of topics affects the results of shared classes and nonmatching classes. Figure 6 shows the box-plots of the number of shared classes and nonmatching classes and shared times in the process of clustering the classes with different numbers of topics. From the results in Figures 6(a) and 6(b), we notice that, with the increasing in the number of topics, the shared counts and the number of shared classes also increase. So setting different values of the number of topics will affect the number of shared classes. In addition, Figure 6(c) shows the results for nonmatching classes in the process of clustering the classes. We see that nonmatching classes are fewer than shared classes. Moreover, the range of the number of nonmatching classes with different numbers of topics is similar. That is, values of different numbers of topics do not obviously affect the number of nonmatching classes.

From the results discussed above, shared classes and nonmatching classes exist in the initial clusters. However, the majority of the classes are neither shared nor nonmatching

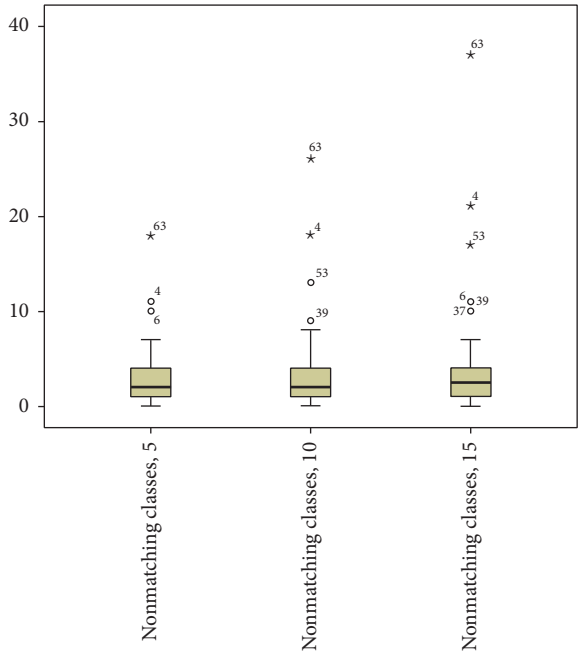
Table 4: The percentage of initial clusters without nonmatching classes and shared classes.

Cluster	$K = 5$ (%)	$K = 10$ (%)	$K = 15$ (%)
Initial clusters without nonmatching classes	90.2	90.1	85.4
Initial clusters without shared classes	70.0	60.0	55.3



(a) The shared counts of different classes in the initial clusters with different number of topics (5, 10, and 15)

(b) The number of shared classes in the initial clusters with different number of topics (5, 10, and 15)



(c) The number of nonmatching classes in the initial clusters with different number of topics (5, 10, and 15)

Figure 6: Shared counts, number of shared classes, and number of nonmatching classes in the initial clusters for number of topics set to 5, 10, and 15.

Table 5: The votes for our approach and LSI-based/PLSA-based clustering approach.

Subject	Package	Our approach	LSI-based	PLSA-based
JHotDraw	*JHotDraw.src.org.jhotdraw.app.action*	7	2	1
jEdit	*jEdit.org.gjt.sp.jedit.gui*	6	1	3
JFreeChart	*jfreechart.source.org.jfree.chart.plot*	5	4	1
muCommander	*muCommander.main.com.mucommander.command*	7	1	2

Table 6: The precision and recall of our approach and LSI-based/PLSA-based clustering approach.

Package	Our approach		LSI-based		PLSA-based	
	P	*R*	*P*	*R*	*P*	*R*
JHotDraw.src.org.jhotdraw.app.action	0.75	0.57	0.44	0.57	0.51	0.43
jEdit.org.gjt.sp.jedit.gui	0.4	0.5	0.04	0.25	0.14	0.22
jfreechart.source.org.jfree.chart.plot	0.83	0.83	1	0.6	0.76	0.68
muCommander.main.com.mucommander.command	0.67	0.33	0.29	0.33	0.42	0.26

ones. Furthermore, some shared classes are shared by three or more topics, and the number of shared classes is larger than that of nonmatching classes. In addition, the results also show that different settings of number of topics will affect the number of shared classes but will not affect the number of nonmatching classes.

5.2. RQ2. In this subsection, we compare the accuracies of the three clustering approaches to show the effectiveness of our approach.

First, we invited participants to assess the clustering results from three clustering approaches. The voting results are shown in Table 5. The results show that, in most cases, the results generated by our approach are more fit to their needs. For the *jfreechart.source.org.jfree.chart.plot* package, the voting results of LSI-based clustering and our approach are similar. When we investigated deep into the results in this package, both two clustering approaches output the clusters with two true-positive relevant classes (the true-positive relevant classes are those classes that do belong to the authoritative cluster). So participants are not sure which one is better than the other one. So from the participants' qualitative analysis, we notice that our approach can generate clustering results which better fit their needs compared to the LSI-based and PLSA-based clustering approach.

In addition, to quantitatively compare these clustering approaches, we compute their precision and recall results, which are shown in Table 6. From the recall perspective, our approach is always better than (or at least as good as) the LSI-based and PLSA-based clustering approaches. However, from the precision perspective, sometimes our results are better, and sometimes the *LSI-based clustering* or *PLSA-based clustering* is better. When we investigate results where our approach achieves lower precision, we notice that the number of classes in the cluster generated by our approach is larger than that by *LSI-based clustering* and *PLSA-based clustering*. For example, for *jfreechart.source.org.jfree.chart.plot* package, there are six classes in the authoritative cluster. Our approach generates five true-positive relevant classes while the *LSI-based clustering* approach generates four and the four are just the true-positive relevant classes. So the precision of the *LSI-based clustering* approach is high while our approach is worse. But from the respective of program comprehension, recall is more important because, with more relevant classes, developers can better comprehend the cluster. That is to say, our approach can cover more relevant classes in the authoritative clusters, which can effectively facilitate program comprehension. So from the results discussed above, compared with *LSI-based clustering* and *PLSA-based clustering*, our approach can effectively identify more relevant classes in a cluster to help program comprehension.

5.3. RQ3. Different from other clustering approaches, our approach also labels each cluster with the topics, which is composed of some words to describe the cluster. In this subsection, we discuss whether these topics are useful to comprehend the cluster.

First, we provided the topics of each cluster to the participants. They used a five-point Likert scale to answer the quality of the topics. The results are shown in Table 7. The average score of the results is around 4, which indicates that the participants think that the topics are useful to understand the cluster. So for program comprehension, the topics labeling the clusters are useful for users to understand the program.

In addition, we also assess the topics of the clusters quantitatively from the precision and recall perspectives. The results are shown in Table 8. For each cluster, our approach can produce a topic which includes some words to label it. These words can cover most of the words given by the participants. For example, for the cluster that includes the *AbstractPieLabelDistributor.java* class, 82% of the words can be covered. Hence, the participants can use these words to help them understand the clusters. In addition, from the precision perspective, our approach is not very good and most of the precision results are about 10%. However, for the other 90% irrelevant words, some are obviously not related

Table 7: The score assessed by the participants on the topics.

Package	Participants										
	P1	*P2*	*P3*	*P4*	*P5*	*P6*	*P7*	*P8*	*P9*	*P10*	*AVG*
JHotDraw.src.org.jhotdraw.app.action	5	5	5	5	4	5	5	4	4	4	4.6
jEdit.org.gjt.sp.jedit.gui	4	4	4	4	3	4	3	4	5	4	3.9
jfreechart.source.org.jfree.chart.plot	3	5	5	4	4	4	4	4	4	4	4.1
muCommander.main.com.mucommander.command	4	4	5	4	4	4	4	3	4	4	4

Table 8: The precision and recall of our approach in inferring representative words to label clusters.

Subject	Package	Class	Topics	
			P	*R*
JHotDraw	*JHotDraw.src.org.jhotdraw.app.action*	*AbstractProjectAction.java*	0.10	0.76
jEdit	*jEdit.org.gjt.sp.jedit.gui*	*AbbrevEditor.java*	0.06	0.80
JFreeChart	*jfreechart.source.org.jfree.chart.plot*	*AbstractPieLabelDistributor.java*	0.10	0.82
muCommander	*muCommander.main.com.mucommander.command*	*AssociationBuilder.java*	0.17	0.64

to the cluster, which are easily identified by the participants, for example, the words "*method,*" "*refer,*" and "*jEdit.*" These words are included in the topics because they are not removed in the preprocessing process. To improve the results, we can improve the preprocessing operations to remove words related to the specific subject programs. Although some noisy information is produced from our approach, the participants still feel that the topics are useful to understand the clusters.

Hence, from the results, we see that the topics in our approach are helpful for developers to understand the clustering results.

6. Threats to Validity

Like any empirical validation, ours has its limitations. In the following, threats to the validity of our case study are discussed.

The first threat relates to the correctness of our experiments and implementation. We have checked the implementation and fixed bugs. Another threat relates to participants' bias. We have reduced this bias by not telling the participants of results produced by our approach and those produced by the baseline approach. In addition, we only applied our technique to four subject programs. Moreover, we considered only one programming language (Java) and one development environment (Eclipse). Further studies are required to generalize our findings to large-scale industrial projects and with developers who have sufficient domain knowledge and familiarity with the subject systems. Thus we cannot guarantee that the results in our case study can be generalized to other more complex or arbitrary subjects. However, these subjects were selected from open-source projects and widely employed for experimental studies [42, 43]. In evaluating the effectiveness of the clustering results, we randomly selected a number of packages. To reduce the threats to validity further, in the future, we plan to evaluate our clustering approach with even more packages from more software projects. The final threat comes from the measures used to evaluate the effectiveness of our approach, that is, precision and recall. These two metrics only focused on the false-positives and false-negatives for authoritative clustering results. However, for program comprehension, other factors may be more important.

7. Related Work

Program comprehension is one of the most important activities in software maintenance and reverse engineering [8, 10, 23, 44, 45]. Clustering techniques are commonly used to decompose a software system into small units for easier comprehension. Some studies analyze syntax features or dependencies to cluster the software [46–50], while others rely on the semantic information in the source code for clustering [51–54].

Clustering approaches based on the syntax (structure) in the source code usually focus on the structural relationships among entities, for example, variable and class references, procedure calls, usage of packages, and association and inheritance relationships among classes. Mancoridis et al. proposed an approach which generates clusters using module dependency graph of the software system [8]. They treated clustering as an optimization problem, which makes use of traditional hill climbing and genetic algorithms. In [46, 55], the Bunch clustering system was introduced. Bunch generates clusters using weighted dependency graph for software maintenance. Sartipi and Kontogiannis presented an interactive approach composed of four phases to recover cohesive subsystems within *C* systems. In the first phase, relations between *C* programs are extracted. In the second phase, these relationships are used to build an attributed relational graph. In the third phase, the graph is manually or automatically partitioned using data mining techniques [56]. These syntax relationships can help developers understand how the functional features are programmed in the source code. In this article, we focus on the clustering based on functional features in the source code. And we used LDA for semantic analysis of these functional features.

Semantic based clustering approaches attempt to show the functional features of a system [57–60]. The functional features in the source code are analyzed from comments, identifier names, and file names [61]. Kuhn et al. presented a language independent approach to group software artifacts based on LSI. They grouped source code containing similar terms in the comments [12, 62]. Scanniello et al. presented an approach to perform the software system partitioning. This approach first analyzes software entities (e.g., programs or classes) and uses LSI to get the dissimilarity between entities, which are grouped by iteratively calling the *K-means* clustering algorithm [63]. Santos et al. used semantic clustering to support remodularization analysis in an input program [58]. Our approach used LDA to generate the clusters, particularly for large-sized packages, to facilitate their comprehension.

In addition, some program comprehension techniques combined the strengths of both syntax and semantic clustering [7, 38, 64–66]. The *ACDC* algorithm is one example of this combined approach which used name and dependency of classes to cluster all classes in a system into small clusters for comprehension [3]. Andritsos and Tzerpos proposed *LIMBO*, a hierarchical algorithm for software clustering [7]. The clustering algorithm considers both structural and nonstructural attributes to reduce the complexity of a software system by decomposing it into clusters. Saeidi et al. proposed to cluster a software system by incorporating knowledge from different viewpoints of the system, that is, knowledge embedded within the source code as well as the structural dependencies within the system, to produce a clustering result [67]. Then, they adopted a search-based approach to provide a multiview clustering of the software system. In this article, we focused on semantic analysis of the source code for its clustering. In addition, our approach also generates topics to help users more easily understand the classes in the clusters.

8. Conclusion and Future Work

In this article, we propose an approach of clustering classes in large-sized packages for program comprehension. Our approach uses LDA to cluster large-sized packages into small clusters, which are labeled with topics to show their features. We conducted case studies to show the effectiveness of our approach on four real-world open-source projects. The results show that the clustering results of our approach are more relevant than those of other clustering techniques, that is, LSI-based and PLSA-based clustering. In addition, the topics labeling these clusters are useful to help developers understand them. Therefore, our approach could provide an effective way for developers to understand large-sized packages quickly and accurately.

In our study, we only conducted studies on four Java-based programs, which does not imply its generality for other types of systems. Future work will focus on conducting more studies on different systems to evaluate the generality of our approach. In addition, during the clustering process, we find that some classes are weakly coupled with its package, but they are more related to another package. That is, there are problems with the current package's structure. So we consider applying our clustering approach to improve the package structure. Finally, our approach is a first step in a top-down program comprehension process; in the future, we plan to cluster other finer-level program elements, for example, methods, to provide a more comprehensive top-down program comprehension support to better understand a software system.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work is supported partially by Natural Science Foundation of China under Grant nos. 61402396, 61472344, 61602267, and 61472343, the Open Funds of State Key Laboratory for Novel Software Technology of Nanjing University under Grant no. KFKT2016B21, the Jiangsu Qin Lan Project, the China Postdoctoral Science Foundation under Grant no. 2015M571489, the Six Talent Peaks Project in Jiangsu Province under Grant no. 2011-DZXX-032, the Natural Science Foundation of the Jiangsu Higher Education Institutions of China under Grant no. 15KJB520030, the Priority Academic Program Development of Jiangsu Higher Education Institutions, and the Jiangsu Collaborative Innovation Center on Atmospheric Environment and Equipment Technology.

References

[1] X. Peng, Z. Xing, X. Tan, Y. Yu, and W. Zhao, "Improving feature location using structural similarity and iterative graph mapping," *Journal of Systems and Software*, vol. 86, no. 3, pp. 664–676, 2013.

[2] J. Wang, X. Peng, Z. Xing, and W. Zhao, "Improving feature location practice with multi-faceted interactive exploration," in *Proceedings of the 35th International Conference on Software Engineering (ICSE '13)*, pp. 762–771, IEEE, San Francisco, Calif, USA, May 2013.

[3] M. P. Obrien, "Software comprehension: a review and research direction," Tech. Rep., 2003.

[4] Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang, "Enabling personalized search over encrypted outsourced data with efficiency improvement," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 9, pp. 2546–2559, 2016.

[5] E. Soloway and K. Ehrlich, "Empirical studies of programming knowledge," *IEEE Transactions on Software Engineering*, vol. 10, no. 5, pp. 595–609, 1984.

[6] W. Maalej, R. Tiarks, T. Roehm, and R. Koschke, "On the comprehension of program comprehension," *ACM Transactions on Software Engineering and Methodology*, vol. 23, no. 4, article 31, 2014.

[7] P. Andritsos and V. Tzerpos, "Information-theoretic software clustering," *IEEE Transactions on Software Engineering*, vol. 31, no. 2, pp. 150–165, 2005.

[8] S. Mancoridis, B. S. Mitchell, C. Rorres, Y. Chen, and E. R. Gansner, "Using automatic clustering to produce high-level system organizations of source code," in *Proceedings of the*

6th International Workshop on Program Comprehension (IWPC '98), p. 45, Ischia, Italy, June 1998.

[9] N. Anquetil and T. Lethbridge, "Experiments with clustering as a software remodularization method," in *Proceedings of the 6th Working Conference on Reverse Engineering (WCRE '99)*, pp. 235–255, IEEE, October 1999.

[10] V. Rajlich and N. Wilde, "The role of concepts in program comprehension," in *Proceedings of the 10th International Workshop on Program Comprehension (IWPC '02)*, pp. 271–278, IEEE, Paris, France, June 2002.

[11] Z. Zhou, Y. Wang, Q. M. Wu, C. Yang, and X. Sun, "Effective and efficient global context verification for image copy detection," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 1, pp. 48–63, 2017.

[12] A. Kuhn, S. Ducasse, and T. Gîrba, "Semantic clustering: identifying topics in source code," *Information & Software Technology*, vol. 49, no. 3, pp. 230–243, 2007.

[13] S. C. Choi and W. Scacchi, "Extracting and restructuring the design of large systems," *IEEE Software*, vol. 7, no. 1, pp. 66–71, 1990.

[14] Y. S. Maarek, D. M. Berry, and G. E. Kaiser, "An information retrieval approach for automatically constructing software libraries," *IEEE Transactions on Software Engineering*, vol. 17, no. 8, pp. 800–813, 1991.

[15] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, no. 4-5, pp. 993–1022, 2003.

[16] X. Sun, X. Liu, B. Li, Y. Duan, H. Yang, and J. Hu, "Exploring topic models in software engineering data analysis: a survey," in *Proceedings of the 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD '16)*, pp. 357–362, IEEE, Shanghai, China, June 2016.

[17] B. Gu, V. S. Sheng, K. Y. Tay, W. Romano, and S. Li, "Incremental support vector learning for ordinal regression," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 7, pp. 1403–1416, 2015.

[18] B. Gu, V. S. Sheng, Z. Wang, D. Ho, S. Osman, and S. Li, "Incremental learning for ν-support vector regression," *Neural Networks*, vol. 67, pp. 140–150, 2015.

[19] J. Tang, Z. Meng, X. Nguyen, Q. Mei, and M. Zhang, "Understanding the limiting factors of topic modeling via posterior contraction analysis," in *Proceedings of the 31th International Conference on Machine Learning*, pp. 190–198, 2014.

[20] A. Panichella, B. Dit, R. Oliveto, M. Di Penta, D. Poshynanyk, and A. De Lucia, "How to effectively use topic models for software engineering tasks? An approach based on genetic algorithms," in *Proceedings of the 35th International Conference on Software Engineering (ICSE '13)*, pp. 522–531, IEEE, May 2013.

[21] Y. Zhang, X. Sun, and B. Wang, "Efficient algorithm for k-barrier coverage based on integer linear programming," *China Communications*, vol. 13, no. 7, pp. 16–23, 2016.

[22] Q. Liu, W. Cai, J. Shen, Z. Fu, X. Liu, and N. Linge, "A speculative approach to spatial-temporal efficiency with multi-objective optimization in a heterogeneous cloud environment," *Security and Communication Networks*, vol. 9, no. 17, pp. 4002–4012, 2016.

[23] D. Binkley, D. Heinz, D. Lawrie, and J. Overfelt, "Understanding LDA in source code analysis," in *Proceedings of the 22nd International Conference on Program Comprehension (ICPC '14)*, pp. 26–36, June 2014.

[24] T. Mens, A. Serebrenik, and A. Cleve, Eds., *Evolving Software Systems*, Springer, 2014.

[25] F. Longo, R. Tiella, P. Tonella, and A. Villafiorita, "Measuring the impact of different categories of software evolution," in *Software Process and Product Measurement, International Conferences: IWSM 2008, Metrikon 2008, and Mensura 2008*, pp. 344–351, 2008.

[26] B. Dit, L. Guerrouj, D. Poshyvanyk, and G. Antoniol, "Can better identifier splitting techniques help feature location?" in *Proceedings of the IEEE 19th International Conference on Program Comprehension (ICPC '11)*, pp. 11–20, IEEE, Ontario, Canada, June 2011.

[27] T. Fritz, G. C. Murphy, E. Murphy-Hill, J. Ou, and E. Hill, "Degree-of-knowledge: modeling a developer's knowledge of code," *ACM Transactions on Software Engineering and Methodology*, vol. 23, no. 2, article 14, 2014.

[28] X. Sun, X. Liu, J. Hu, and J. Zhu, "Empirical studies on the NLP techniques for source code data preprocessing," in *Proceedings of the 3rd International Workshop on Evidential Assessment of Software Technologies (EAST '14)*, pp. 32–39, May 2014.

[29] G. Santos, M. T. Valente, and N. Anquetil, "Remodularization analysis using semantic clustering," in *Proceedings of the Software Evolution Week—IEEE Conference on Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE '14)*, pp. 224–233, Antwerp, Belgium, February 2014.

[30] T. Hofmann, "Unsupervised learning by probabilistic latent semantic analysis," *Machine Learning*, vol. 42, no. 1-2, pp. 177–196, 2001.

[31] T. Hofmann, "Probabilistic latent semantic analysis," in *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI '99)*, pp. 289–296, Stockholm, Sweden, July 1999, https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&proceeding_id=15 &article_id=179.

[32] Y. Liu, D. Poshyvanyk, R. Ferenc, T. Gyimóthy, and N. Chrisochoides, "Modeling class cohesion as mixtures of latent topics," in *Proceedings of the IEEE International Conference on Software Maintenance (ICSM '09)*, pp. 233–242, Alberta, Canada, September 2009.

[33] M. Shtern and V. Tzerpos, "Clustering methodologies for software engineering," *Advances in Software Engineering*, vol. 2012, Article ID 792024, 18 pages, 2012.

[34] M. P. Robillard and G. C. Murphy, "Representing concerns in source code," *ACM Transactions on Software Engineering and Methodology*, vol. 16, no. 1, article 3, 2007.

[35] D. Binkley, M. Ceccato, M. Harman, F. Ricca, and P. Tonella, "Tool-supported refactoring of existing object-oriented code into aspects," *IEEE Transactions on Software Engineering*, vol. 32, no. 9, pp. 698–717, 2006.

[36] S. Deerwester, "Improving information retrieval with latent semantic indexing," in *Proceedings of the Annual Meeting of the American Society for Information Science*, pp. 1–10, 1988.

[37] D. Poshyvanyk, M. Gethers, and A. Marcus, "Concept location using formal concept analysis and information retrieval," *ACM Transactions on Software Engineering and Methodology*, vol. 21, no. 4, pp. 1–34, 2012.

[38] J. I. Maletic and A. Marcus, "Supporting program comprehension using semantic and structural information," in *Proceedings of the 23rd International Conference on Software Engineering*, pp. 103–112, May 2001.

[39] J. Han, *Data Mining: Concepts and Techniques*, Morgan Kaufmann, San Francisco, Calif, USA, 2005.

[40] X. Sun, B. Li, Y. Li, and Y. Chen, "What information in software historical repositories do we need to support software maintenance tasks? An approach based on topic model," in *Computer and Information Science*, pp. 27–37, Springer International Publishing, 2015.

[41] C. J. van Rijsbergen, *Information Retrieval*, Butterworths, London, UK, 1979.

[42] U. Erdemir, U. Tekin, and F. Buzluca, "Object oriented software clustering based on community structure," in *Proceedings of the 18th Asia Pacific Software Engineering Conference (APSEC '11)*, pp. 315–321, IEEE, Ho Chi Minh, Vietnam, December 2011.

[43] A. De Lucia, M. Di Penta, R. Oliveto, A. Panichella, and S. Panichella, "Using IR methods for labeling source code artifacts: is it worthwhile?" in *Proceedings of the 20th IEEE International Conference on Program Comprehension (ICPC '12)*, pp. 193–202, June 2012.

[44] X. Liu, X. Sun, B. Li, and J. Zhu, "PFN: a novel program feature network for program comprehension," in *Proceedings of the 13th IEEE/ACIS International Conference on Computer and Information Science (ICIS '14)*, pp. 349–354, Taiyuan, China, June 2014.

[45] Z. Fu, X. Wu, C. Guan, X. Sun, and K. Ren, "Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 12, pp. 2706–2716, 2016.

[46] B. S. Mitchell and S. Mancoridis, "On the automatic modularization of software systems using the bunch tool," *IEEE Transactions on Software Engineering*, vol. 32, no. 3, pp. 193–208, 2006.

[47] S. Islam, J. Krinke, D. Binkley, and M. Harman, "Coherent clusters in source code," *Journal of Systems and Software*, vol. 88, no. 1, pp. 1–24, 2014.

[48] S. Mirarab, A. Hassouna, and L. Tahvildari, "Using Bayesian belief networks to predict change propagation in software systems," in *Proceedings of the 15th IEEE International Conference on Program Comprehension (ICPC '07)*, pp. 177–186, June 2007.

[49] F. Deng and J. A. Jones, "Weighted system dependence graph," in *Proceedings of the 5th IEEE International Conference on Software Testing, Verification and Validation (ICST '12)*, pp. 380–389, Montreal, Canada, April 2012.

[50] M. Gethers, A. Aryani, and D. Poshyvanyk, "Combining conceptual and domain-based couplings to detect database and code dependencies," in *Proceedings of the IEEE 12th International Working Conference on Source Code Analysis and Manipulation (SCAM '12)*, pp. 144–153, IEEE, Trento, Italy, September 2012.

[51] Z. Fu, X. Sun, Q. Liu, L. Zhou, and J. Shu, "Achieving efficient cloud search services: multi-keyword ranked search over encrypted cloud data supporting parallel computing," *IEICE Transactions on Communications*, vol. E98B, no. 1, pp. 190–200, 2015.

[52] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 340–352, 2016.

[53] L. Guerrouj, "Normalizing source code vocabulary to support program comprehension and software quality," in *Proceedings of the 35th International Conference on Software Engineering (ICSE '13)*, pp. 1385–1388, San Francisco, Calif, USA, May 2013.

[54] A. De Lucia, M. Di Penta, and R. Oliveto, "Improving source code lexicon via traceability and information retrieval," *IEEE Transactions on Software Engineering*, vol. 37, no. 2, pp. 205–227, 2011.

[55] N. Anquetil and T. C. Lethbridge, "Recovering software architecture from the names of source files," *Journal of Software Maintenance and Evolution*, vol. 11, no. 3, pp. 201–221, 1999.

[56] K. Sartipi and K. Kontogiannis, "A user-assisted approach to component clustering," *Journal of Software Maintenance and Evolution*, vol. 15, no. 4, pp. 265–295, 2003.

[57] T. Ma, J. Zhou, M. Tang et al., "Social network and tag sources based augmenting collaborative recommender system," *IEICE Transactions on Information and Systems*, vol. E98-D, no. 4, pp. 902–910, 2015.

[58] G. Santos, M. T. Valente, and N. Anquetil, "Remodularization analysis using semantic clustering," in *Proceedings of the 1st Software Evolution Week—IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE '14)*, pp. 224–233, February 2014.

[59] Z. Xia, X. Wang, X. Sun, and B. Wang, "Steganalysis of least significant bit matching using multi-order differences," *Security and Communication Networks*, vol. 7, no. 8, pp. 1283–1291, 2014.

[60] S. Kawaguchi, P. K. Garg, M. Matsushita, and K. Inoue, "Mudablue: an automatic categorization system for open source repositories," in *Proceedings of the 11th Asia-Pacific Software Engineering Conference (APSEC '04)*, pp. 184–193, Busan, Republic of Korea, December 2004.

[61] A. Kuhn, S. Ducasse, and T. Gîrba, "Enriching reverse engineering with semantic clustering," in *Proceedings of the 12th Working Conference on Reverse Engineering (WCRE '05)*, pp. 133–142, Pittsburgh, Pa, USA, November 2005.

[62] A. Corazza, S. Di Martino, V. Maggio, and G. Scanniello, "Investigating the use of lexical information for software system clustering," in *Proceedings of the 15th European Conference on Software Maintenance and Reengineering (CSMR '11)*, pp. 35–44, IEEE, Oldenburg, Germany, March 2011.

[63] G. Scanniello, M. Risi, and G. Tortora, "Architecture recovery using Latent Semantic Indexing and k-Means: an empirical evaluation," in *Proceedings of the 8th IEEE International Conference on Software Engineering and Formal Methods (SEFM '10)*, pp. 103–112, September 2010.

[64] G. Scanniello, A. D'Amico, C. D'Amico, and T. D'Amico, "Using the Kleinberg algorithm and vector space model for software system clustering," in *Proceedings of the 18th IEEE International Conference on Program Comprehension (ICPC '10)*, pp. 180–189, IEEE, Braga, Portugal, June-July 2010.

[65] G. Scanniello and A. Marcus, "Clustering support for static concept location in source code," in *Proceedings of the IEEE 19th International Conference on Program Comprehension (ICPC '11)*, pp. 36–40, Kingston, Canada, June 2011.

[66] Y. Kong, M. Zhang, and D. Ye, "A belief propagation-based method for task allocation in open and dynamic cloud environments," *Knowledge-Based Systems*, vol. 115, pp. 123–132, 2017.

[67] A. M. Saeidi, J. Hage, R. Khadka, and S. Jansen, "A search-based approach to multi-view clustering of software systems," in *Proceedings of the 22nd IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER '15)*, pp. 429–438, March 2015.

Hindawi
Scientific Programming
Volume 2017, Article ID 5159690, 9 pages
https://doi.org/10.1155/2017/5159690

Research Article

Exploring the Evolution of New Mobile Services

Yong Chen,[1,2] Juncheng Yao,[1] Hai Jin,[1] Chunjiang He,[2] and Hanhua Chen[1]

[1]*Big Data Technology and System Lab, Services Computing Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China*
[2]*China Electric Power Research Institute, Beijing 100192, China*

Correspondence should be addressed to Hai Jin; hjin@hust.edu.cn

Received 6 January 2017; Accepted 9 March 2017; Published 6 April 2017

Academic Editor: Alex M. Kuo

The emergence and widespread use of mobile Internet technology has led to many different kinds of new mobile communications services, such as *WeChat*. Users could have more choices when attempting to satisfy their communications needs. The ability to predict the way in which users will use new mobile communications services is extremely valuable to mobile communications service providers. In this work, we propose a method for predicting how a user will use a new mobile service. Our scheme is inspired by the evolutionary game theory. With large-scale real world datasets collected from mobile service providers, we first extract the benefit-related features for users who were starting to use a new mobile service. Then we design our training and prediction methods for predicting potential users. We evaluate our scheme using experiments with large-scale real data. The results show that our approach can predict users' future behavior with satisfying accuracy.

1. Introduction

With the development of mobile Internet technology [1–3] and large-scale distributed systems [4, 5], many new mobile services are being offered by a diverse range of providers [6, 7], including traditional service providers and Internet service providers. Communications between users through the data domain over an Internet connection are known as "Over the Top (OTT)" communications. These new kinds of mobile services provide users with a greater range of choices. Tencent provides communications services through *WeChat* and *QQ*, while as a result of the development of 4G technology, China Mobile starts to provide communications services through VoLTE technology. For example, as the most popular OTT communications service, *WeChat* had about 8.2 million monthly active users as of October 2016. More than 60% of *WeChat* users open *WeChat* more than 10 times per day. *WeChat* users post about 500 million messages every day. Competition between these service providers is becoming increasingly intense.

Users can choose appropriate mobile communications services according to their preferences [8, 9]. Being able to predict the way in which users will likely use a new mobile communications service would be extremely useful to mobile communications service providers. This would help service providers understand consumers' demands and requirements, provide personalized services according to a user's preferences, and improve product features and competition policy. Therefore, a method for predicting the way in which users would use a new mobile communications service, in an accurate and timely manner, has become a topic of great concern to service providers.

There have been many studies that have focused on predicting the behavior of a user of a mobile service. Some researchers have studied the reasons and factors influencing the users of a mobile service. Some research works have studied the prediction of app use in a mobile phone. Some methods for predicting the way in which a user uses a mobile service have been proposed. However, these methods cannot accurately predict the way in which users will use a new mobile service. Mainstream companies like Google and Apple use signal models considering session duration, operations per session, location, and device data to predict mobile service usage. In order to predict users behavior with new mobile service, we have added network side data, refined the data usage, and improved the forecasting model.

In this study, we first analyze the characteristics of a users' behavior and propose our concept for predicting how a user will make use of a new service. We propose a model for predicting a user's behavior when using a new mobile communications service based on evolutionary game theory. We extract features through the perspective of users' motivation when using a new mobile service, based on users' behavioral data, training the model to predict the potential users and heavy users of a new service. Then, we classify the users into different groups according to the users' features, training the model for each group of users to obtain a better performance. We evaluated the performance of our method with real users' usage behavior datasets. The results showed that our approach can predict the users' future usage behavior to attain a high level of accuracy in a relatively short time.

This paper is organized as follows. In Section 2, we describe other studies related to mobile service prediction. In Section 3, we define the problem being addressed in detail. In Section 4, we illustrate our ideas and methods for predicting users' behavior regarding a new mobile service. Section 5 describes the experiments used to validate our model. Finally, we summarize our work in Section 6.

2. Related Work

A better understanding of users' behavior is of great significance. Since the mobile instant messaging service became an important means of communication, researchers have undertaken many interesting studies related to the analysis of users' behavior. For example, Oghuma et al. [10] studied users' ongoing intentions to use mobile instant messaging. By applying an expectation confirmation model, they pointed out that the service quality will affect users satisfaction significantly, and they [11] explained a new user's ongoing intention to use mobile instant messaging applications with a benefit-confirmation model. Ogara et al. [12] studied the main factors that influence a user's satisfaction with a mobile messaging service. They showed that experience and social factors are the main influences on user satisfaction. Kuo and Yen [13] studied consumer's behavioral intentions to adopt mobile value-added services. Hong et al. [14] compared the efficiency of expectation confirmation model and acceptance model for understanding the ongoing service usage behavior. The data used in these studies were collected from a user survey, such that these data directly reflect users' intentions.

Many previous works studied users' behavior as it is related to mobile apps [15]. Sun et al. [16] studied the mobile usage prediction problem using feature comparison analysis. To realize a dynamic home screen application, Shin et al. [17] built a prediction model [18, 19] for the usage of apps in mobile context. Zhu et al. [20] studied service usage patterns from the perspective of users and proposed an application usage prediction method. Liao et al. [21] predicted app usage through the application of a temporal feature model, and they [22] proposed a method for predicting which apps are most likely to be used with a temporal-based model. Xu [23] proposed a method for predicting Chinese mobile user behavior based on a time sequence analysis. Kloumann et al. [24] analyzed the temporal and social profiles in app usage prediction problem. These methods of predicting mobile application usage, based on data collected from telephone handsets, are extremely valuable for predicting the way in which an app is used [25–29].

Some studies have focused on the prediction of users' usage of mobile services. Haddad et al. [30] proposed a means of predicting periodic consumption behaviors based on a poison processes model. Valera and Gomez-Rodriguez [31] developed a method to predict the adoption and use frequency of similar products in social networks.

Our work differs from the previous studies. The motivation behind our work involves the prediction of the users' behavior when a new mobile service is offered on the market. It is difficult to predict the users' usage behavior from the usage trace data since the amount of available usage trace data is insufficient. The data used for prediction in our work was obtained from mobile service providers, so how to make full use of these data to determine a user's motivation to use a new service is the key issue in our work. Our concept was inspired by dynamic game theory [32–34]. Taylor and Jonker [35] studied the evolution of the game in dynamic case and described the state of the system by differential equations. Goyal et al. [36] proposed a game-theoretic framework to study the initial adoption of competing products.

3. Problem Definition

The main aim of this work was to predict whether a user will continue to use a new mobile service after that user starts to use it and whether a user will become a high-frequency user of the new mobile service. We call a user who will continue to use a new mobile service a "potential user" of the new service, while a user who will use a new service at a high frequency is a "heavy user" of the new service. The datasets collected from a service provider contain the activities of each user u using service s at time t. From these datasets, we can determine the use times $\mathrm{UT}(u, s, t)$ and data flow $\mathrm{DF}(u, s, t)$ for a user u using service s over a period t. When we combine the two parameters to give the level $\mathrm{UL}(u, s, t)$ for user u using service s over a period t, the equation for the use level will be

$$\mathrm{UL}(u, s, t) = \lambda\alpha\mathrm{UT}(u, s, t) + (1 - \lambda)\beta\mathrm{DF}(u, s, t), \quad (1)$$

where λ is the weight factor and α and β are the normalization factors. When the duration from the first time u uses new service s_n is sufficiently long, the $\mathrm{UL}(u, s_n, t)$ can be the continuous use level $\mathrm{CUL}(u, s_n)$. For user u using a new service s_n, the equation of the continuous use level is

$$\mathrm{CUL}(u, s_n) = \mathrm{UL}(u, s_n, t) \quad t \gg t_0(u, s_n), \quad (2)$$

where $t_0(u, s_n)$ is the first time user u uses service s.

We first need to identify the potential and heavy users from our training dataset. We extract the user's usage data for the period starting when the user first uses the new mobile service up until the point when the user has had sufficient time to become familiar with the new service. We then apply

the potential user label L_{PU} and heavy user label L_{HU} by setting a use level threshold

$$L_{\mathrm{PU}} = 0 \quad \text{when } \mathrm{CUL}(u, s_n) < \mathrm{threshold}_{\mathrm{PU}}, \tag{3a}$$

$$L_{\mathrm{PU}} = 1 \quad \text{when } \mathrm{CUL}(u, s_n) \geq \mathrm{threshold}_{\mathrm{PU}}, \tag{3b}$$

$$L_{\mathrm{HU}} = 0 \quad \text{when } \mathrm{CUL}(u, s_n) < \mathrm{threshold}_{\mathrm{HU}}, \tag{4a}$$

$$L_{\mathrm{HU}} = 1 \quad \text{when } \mathrm{CUL}(u, s_n) \geq \mathrm{threshold}_{\mathrm{HU}}. \tag{4b}$$

Subsequently, we can apply a label to a user u according on whether that user is a potential user or a heavy user. The problem we want to solve is how to predict if a user will be a potential user label or a heavy user in the beginning when a user starts to use the new service. This is a classification problem, allowing us to predict the probability of a user u continuing to use a new mobile service, that is, the probability of potential user P_{PU} and the probability of a user u being a high-frequency user of the new mobile service, that is, the probability of heavy user P_{HU}.

The datasets we use are collected from service providers. These data include users' behavior and users' connection technical parameters for the new service.

4. Methodology

In this section, we first present the concept of predicting users' usage behavior for new mobile communications services. Then, we introduce a method for addressing the prediction problem, including feature extraction and the prediction method.

4.1. Prediction Model. We need to predict a users' feature usage behavior in an accurate and timely manner using data collected from service providers. It is difficult to predict users' feature usage behavior in the initial stages when a user is only beginning to use a new mobile service, since there is insufficient usage behavior data.

The intuitive concept on which our method is based is that a user's motivation leading to his or her use behavior can be explained by evolutionary game theory. Evolutionary game theory studies how biological species choose their dominating strategy and make better decisions. The basic concept is that biological species will make decisions when faced with many options according to their perceived strategy benefits. When some individuals of a species attempt a new strategy, they will evaluate the benefits of the new strategy. If the benefits of the new strategy are greater than those of an alternative strategy, there will be a greater probability of that strategy being used again. When the mean benefit of the new strategy for one species is greater than the mean benefit for all the species, the new strategy will become an evolutionary stable strategy, and the ratio of the new strategy in the species will gradually increase among the features.

Faced with this problem, the user will evaluate the new service's perceived benefits based on his or her experience when they first try to use the service, and the benefits will affect the user's usage behavior for the new service. Therefore, we can predict users' usage behavior from the users' motivation perspective. We analyze the data obtained from the service providers and extract the features from the users' experience and benefit-related data.

4.2. Feature Extraction. By studying the main factors influencing the user's experience and benefits of mobile services as addressed in related works, we identified four categories of factors that greatly influence the user's experience and benefits, including the service technology factors, the ability of the services factors, interaction factors, and cost factors. The notations in feature extraction [37, 38] are listed in Notations.

4.2.1. Service Technology Feature. The service technology feature in a mobile service including the end-to-end delay, connection rate, and transmission speed will have a direct impact on the users' experience when using a mobile communications service. From the users' connection technical datasets, we can extract the service technology parameter, after which we transform these parameters to a normalized feature to indicate the service technology feature including the end-to-end delay feature, connected rate feature, and transmit speed feature for each user when using the new service. These features are counted over a period of time, typically for one day. The features description of service technology is listed in Table 1.

The end-to-end delay feature is estimated by the average end-to-end delay, each time user u uses service s in time period t

$$\mathrm{ED}(u, s, t) = \frac{\sum_{i \in t} \mathrm{ED}_i(u, s)}{\sum_{i \in t} \mathrm{UC}_i(u, s)}. \tag{5}$$

The connect rate feature is estimated by the average connect rate, each time user u uses the service s in time period t

$$\mathrm{CR}(u, s, t) = \frac{\sum_{i \in t} \mathrm{CR}_i(u, s)}{\sum_{i \in t} \mathrm{UC}_i(u, s)}. \tag{6}$$

The transmit speed feature is estimated by the average transmit speed for each time user u uses the service s in time period t

$$\mathrm{TS}(u, s, t) = \frac{\sum_{i \in t} \mathrm{TS}_i(u, s)}{\sum_{i \in t} \mathrm{UC}_i(u, s)}. \tag{7}$$

4.2.2. Service Ability Feature. The service ability feature indicates the benefit a user can derive from a new service. We compare the new service usage to other services of the same type and then estimate the service ability feature of the new service. These features are counted over a period of time, typically for one day. The features description of service ability is listed in Table 2.

The use frequency feature is estimated by summing the use time for user u, using service s in time period t

$$\mathrm{UF}(u, s, t) = \sum_{i \in t} \mathrm{UC}_i(u, s). \tag{8}$$

Table 1: Service technology feature description.

Feature	Description
End_to_end_delay (ED)	Average end-to-end delay for user u in time period t when using the new service s
Connect_rate (CR)	Average connect rate for user u in time period t when using the new service s
Transmit_speed (TS)	Average transmit speed for user u in time period t when using the new service s

Table 2: Service ability feature description.

Feature	Description
Use_frequency (UF)	New service s use count for user u during time period t
Use_amount (UA)	New service s use amount for user u during time period t
Use_frequency_ratio (UFR)	Ratio of the new service s use count over all service of same category for user u during time period t
Use_amount_ratio (UAR)	Ratio of the new service s using amount over all service of same category for user u during time period t

The use amount is estimated by summing the use count for user u using service s over time period t

$$\mathrm{UA}(u,s,t) = \sum_{i\in t} \mathrm{UA}_i(u,s). \tag{9}$$

The use frequency ratio feature is estimated from the ratio of the service s use time over all the services of the same category use time for user u in time period t

$$\mathrm{UFR}(u,s,t) = \frac{\mathrm{UF}(u,s,t)}{\sum_{s\in S} \mathrm{UF}(u,s,t)}. \tag{10}$$

The use amount ratio feature is estimated from the ratio of the service s use amount over all the services of the same category use amount for user u in time period t

$$\mathrm{UAR}(u,s,t) = \frac{\mathrm{UA}(u,s,t)}{\sum_{s\in S} \mathrm{UA}(u,s,t)}. \tag{11}$$

4.2.3. Interaction Feature. The interaction feature indicates the usability of a new service for a user. We can extract the adaptation time needed for a user to become familiar with a new service and the ratio of the use amount during the learning period over the mean learning amount [39]. The adaptation time feature is extracted from a user's use behavior by calculating the time between the first use to the achievement of stable use. The learning ratio feature will be extracted from a user's use amount during the learning time over the mean learning amount. The features description of interaction is listed in Table 3.

The learning time feature is estimated from the time between user u starting to use service s and the use level of service s reaching a multiple of the long period use level corresponding to the learning level

$$\begin{aligned} \mathrm{LT}(u,s) = \min \quad & (t_s) - t_0, \\ \text{s.t.} \quad & \mathrm{UL}(u,s,t_s) > \gamma \frac{\sum_{t\in T} \mathrm{UL}(u,s,t)}{T(u,s)}, \end{aligned} \tag{12}$$

where γ is the learning level parameter.

The learning ratio feature is estimated as a summation of the use amount during the period needed for user u to use service s

$$\mathrm{LR}(u,s) = \sum_{t\in \mathrm{LT}(u,s)} \mathrm{UA}(u,s,t). \tag{13}$$

4.2.4. Cost Feature. The cost feature indicates the cost of using a new service. We extract the cost of a new service for each time or amount of use. The cost feature is a relatively stable feature for a certain new service over a short period of time. These features are counted over a period of time, typically for one day. We can get this data from the service provider. The features description of cost is listed in Table 4.

The time cost feature is estimated as the mean cost of each use of service s by user u in time period t

$$\mathrm{TC}(u,s,t) = \frac{\sum_{i\in t} \mathrm{TC}_i(u,s)}{\sum_{i\in t} \mathrm{UC}_i(u,s)}. \tag{14}$$

The amount cost feature is estimated as the mean cost for the total amount user u uses service s in time period t.

$$\mathrm{AC}(u,s,t) = \frac{\sum_{i\in t} \mathrm{AC}_i(u,s)}{\sum_{i\in t} \mathrm{UA}_i(u,s)}. \tag{15}$$

These features are extracted from the use behavior and connection technical parameter datasets for each user. They will indicate each user's experience directly, and from these data we can precisely predict the use behavior for each user.

4.3. Prediction Method. In the prediction step, we first employ a pretraining part to train the benefits of different feature categories using regression, and then we train the users' use behavior data with the benefits of four categories. We design a classification part to separate a different kind of user to promote the performance of the prediction model.

4.3.1. Pretraining. To train the benefit of four categories of features, we implement a pretraining step for the four categories of features. The target value of this pretraining

Table 3: Interaction feature description.

Feature	Description
Learning_time (LT)	Learning time for user u when user first using the new service s
Learning_ratio (LR)	Learning ratio for user u when user first uses the new service s

Table 4: Cost feature description.

Feature	Description
Time_cost (TC)	Mean cost of each use of a new service s for user u in period t
Amount_cost (AC)	Mean cost for total use of new service s for user u in period t

model will be the users' long-term use level, which can indicate the benefits of the user to some degree. Therefore, the function we want to train is the benefit factor for the four categories. The training model can be a regression model. After the pretraining step, we can check whether the parameter of each feature is reasonable, test the efficacy of the feature, and guarantee the correctness of our model; then the model will be less error-prone

$$\begin{aligned} &\mathrm{CUL}(u,s) \sim E_{\mathrm{TECH}}(u,s) = f_{\mathrm{TECH}}(\mathrm{ED},\mathrm{CR},\mathrm{TS}), \\ &\mathrm{CUL}(u,s) \sim E_{\mathrm{ABIL}}(u,s) \\ &\qquad = f_{\mathrm{ABIL}}(\mathrm{UF},\mathrm{UA},\mathrm{UFR},\mathrm{UAR}), \\ &\mathrm{CUL}(u,s) \sim E_{\mathrm{INTE}}(u,s) = f_{\mathrm{INTE}}(\mathrm{LT},\mathrm{LR}), \\ &\mathrm{CUL}(u,s) \sim E_{\mathrm{COST}}(u,s) = f_{\mathrm{COST}}(\mathrm{TC},\mathrm{AC}). \end{aligned} \tag{16}$$

4.3.2. Prediction of Potential and Heavy Users. Our goal is to predict whether a user will continue to use a new mobile service after he or she starts to use it and whether a user is a high-frequency user of the new mobile service. Once we know the benefits for each user when they are trained in the use of the new mobile service in the pretraining part, we can calculate the benefits of the four categories. Then, we can get the users' perceived benefit of the four factors for each user. Since the users' use probability distribution is a sigmoid form conforming to evolutionary game theory, we use a logistic regression model to train the potential user with the users' label data we have identified. Figure 1 shows the simulated use probability variation with relative perceived benefit by using evolutionary game theory. Heavy user prediction is done using the same method as that for a potential user

$$\begin{aligned} L_{\mathrm{PU}}(u,s) &\sim f_{\mathrm{PU}}(E_{\mathrm{TECH}},E_{\mathrm{ABIL}},E_{\mathrm{INTE}},E_{\mathrm{COST}}), \\ L_{\mathrm{HU}}(u,s) &\sim f_{\mathrm{HU}}(E_{\mathrm{TECH}},E_{\mathrm{ABIL}},E_{\mathrm{INTE}},E_{\mathrm{COST}}). \end{aligned} \tag{17}$$

4.3.3. Prediction of Potential and Heavy Users after Classification of User Groups. The preference of different users is different; the above method neglects the difference between users. We classify the users into several groups; then we train the above model for each group. The parameters of each group may be different from each other. The predicted results will be compared in the evaluation section. The features we use for classification can be part of the features we extracted from users in Section 4.2; other features such as the city of the user or the age of the user can be used for classification. After classification, we can predict a users' usage behavior for different groups of users.

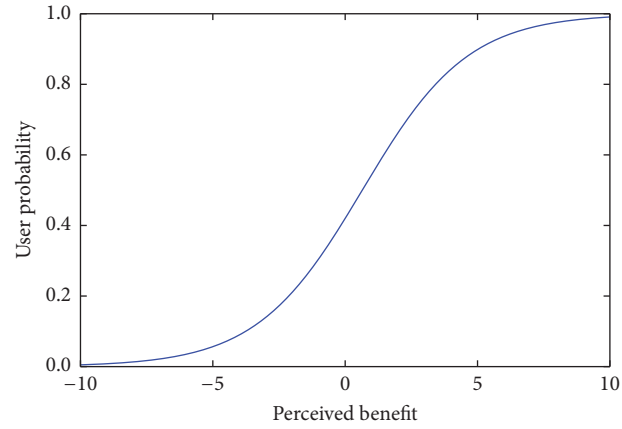


Figure 1: Simulated use probability variation with relative perceived benefit.

5. Performance Evaluation

5.1. Dataset. In order to verify the performance of this method, we collected users' usage behavior data for 1,048,574 users of China Mobile Communications Corporation in 14 cities across China. The date range of these data was from September 2012 to March 2013. The new mobile service covered by the dataset was *WeChat*. We collected the users' behavior record, users' technical information, and users' service plan data from operator side. After checking the data integrity, we obtained 330,331 users' data for training and testing. We identified the potential and heavy users of *WeChat* from the last two weeks of our users' behavior data. We found 238,040 potential users and 90,905 heavy users of *WeChat*. The statistical information [40] of samples is listed in Table 5.

5.2. Performance of the Proposed Approach. We extracted the feature data for the three weeks from when a user first uses *WeChat*. After the pretraining step, we examined the efficacy of each feature and then trained the logistic regression model

Table 5: Statistical information of samples.

	Potential user		Heavy user	
	Label 1	Label 0	Label 1	Label 0
Numbers	238040	92291	90905	239426
Percentage	72.06%	27.94%	27.52%	72.48%

Table 6: Performance of two models.

Model	Precision	Recall	$F1$	AUC
1-class potential user	0.949	0.939	0.943	0.911
2-class potential user	0.952	0.959	0.956	0.927
Baseline potential user	0.844	0.802	0.829	0.801
1-class heavy user	0.501	0.805	0.617	0.872
2-class heavy user	0.506	0.811	0.623	0.882
Baseline heavy user	0.413	0.751	0.533	0.747

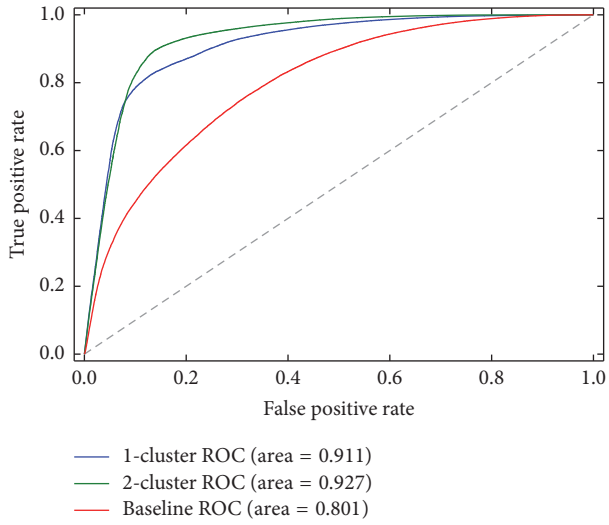


Figure 2: Comparison between ROC curves for potential user prediction.

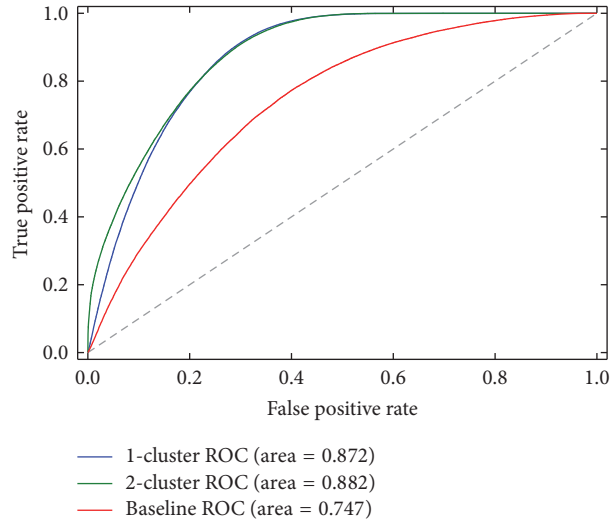


Figure 3: Comparison between ROC curves for heavy user prediction.

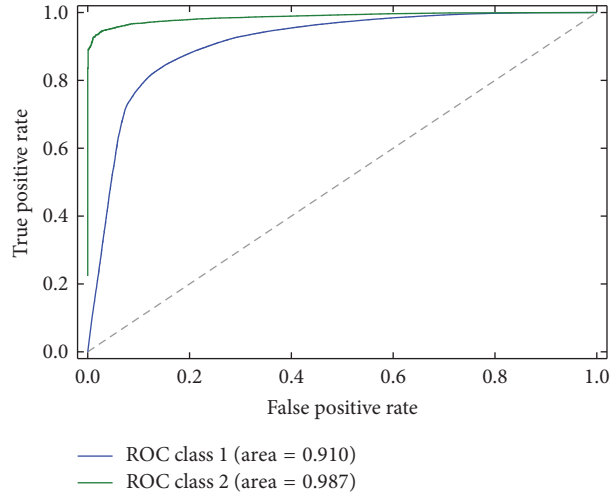


Figure 4: ROC curve for each class of 2-class potential user prediction.

for potential and heavy users. Then, we divided the users into several groups and retrained the model to make a comparison with the nonclassified model. The baseline model uses a signal model trained with duration and frequency data of the three-week users' behavior which is used as a comparison of this experiment.

We used the precision, recall, $F1$ score, and AUC score of ROC to measure the models' performance [41]. First, we divided the users into two groups according to the classification method described in prediction method and then retrained the parameters for each group. Then, we compared the model's performance to the nonclassified model and the baseline model. Table 6 shows the results, where 1-class indicates the nonclassified model and 2-class indicates the model for which two classes are classified.

Then, we compared the ROC curve for the nonclassified users, two classes of classified users, and the baseline model. Figure 2 shows the comparison between ROC curves for potential user prediction. Figure 3 shows the comparison between ROC curves for heavy users.

We compared the ROC curve for different classes. Figure 4 shows the ROC curve for each of two classes classified as potential users. Figure 5 shows the ROC curve for each class of three classes classified as potential users. Figure 6 shows the ROC curve for each class of two classes classified as heavy users. Figure 7 shows the ROC curve for each class of three classes classified as heavy users.

We used a 5-folder cross-validation method to evaluate the performance of the models. The ROC curve for each fold was compared. Figure 8 shows the ROC curve for each fold in nonclassified potential user prediction. Figure 9 shows the ROC curve for each fold in nonclassified heavy user prediction.

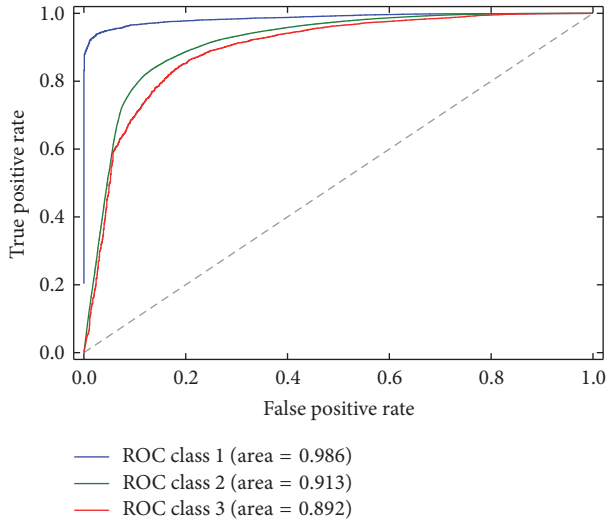


Figure 5: ROC curve for each class in 3-class potential user prediction.

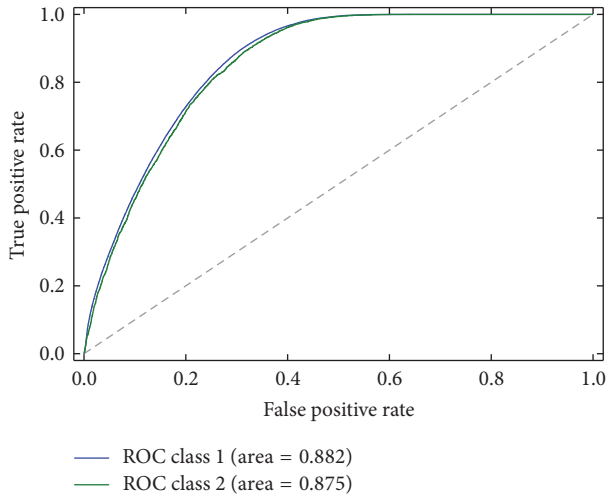


Figure 6: ROC curve for each class of 2-class heavy user prediction.

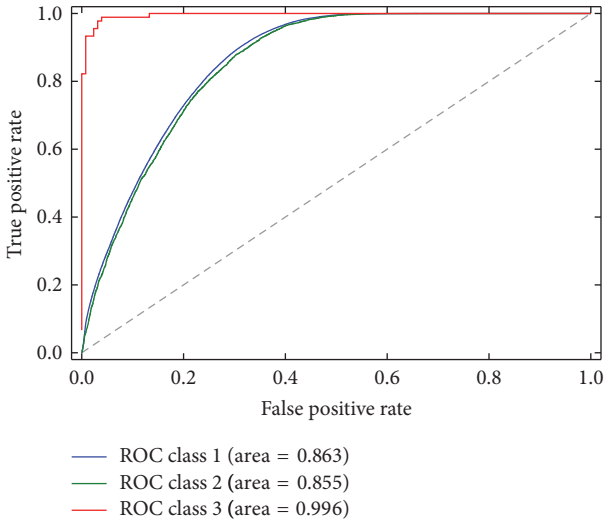


Figure 7: ROC curve for each class in 3-class heavy user prediction.

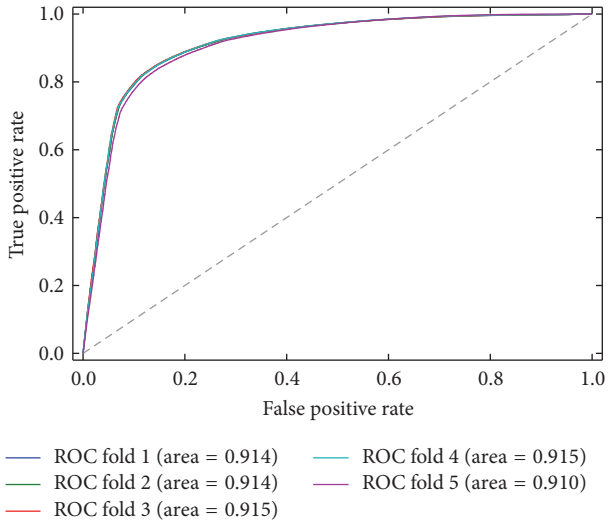


Figure 8: ROC curve for each fold in 1-class potential user prediction.

From the evaluation results, we can find that our models for potential and heavy user prediction offer a better performance compared with the baseline model. The prediction of potential users exhibits a higher level of performance since the potential users of WeChat constitute a larger proportion of all the users, so the prediction is easier. When we classify the users into two groups, the performance for each group will be better in that the mean performance will increase by 1% to 2%. When we divide the users into more groups, the performance will improve by 1% to 3%, but the gain difference will be small. The classes of user performance differ from each other, since the classification method can gather similar users together to make it easier to predict. The difference between the different folds is relatively small, indicating that the model is very stable.

6. Conclusion

Inspired by the evolutionary game theory, in this paper we have proposed a novel scheme for predicting users' usage of new mobile services. We set out to understand the users' experience of a new mobile service by extracting the benefit-related features from the users' behavior data. We then designed a training and prediction method for potential and heavy users. We classified the users into several groups to distinguish between users for group user training. Finally we evaluated our approach using comprehensive experiments with data from real world systems. The results showed that our approach can predict the users' future usage behavior with a high level of performance in a relatively short time. We hope that the concept addressed by our study in this work will help future research in this direction.

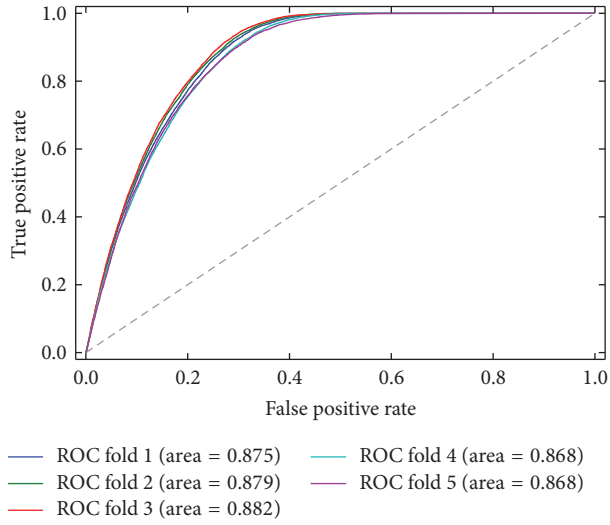


Figure 9: ROC curve for each fold in 1-class heavy user prediction.

Notations

$UC_i(u, s)$: Service use count for user u using service s at time i
$ED_i(u, s)$: End-to-end delay for user u using service s at time i
$CR_i(u, s)$: Connect rate for user u using service s at time i
$TS_i(u, s)$: Transmit speed for user u using service s at time i
S: All service sets
$T(u, s)$: All time period sets for user u using service s
$TC_i(u, s)$: Time cost for user u using service s at time i
$AC_i(u, s)$: Cost for user u to use service s at time i
$UA_i(u, s)$: Service use amount for user u using service s at time i.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research is supported in part by the National Key Research and Development Program of China under Grant no. 2016QY02D0202, NSFC under Grants nos. 61370233, 61422202, and 61433019, Foundation for the Author of National Excellent Doctoral Dissertation of China under Grant no. 201345, and Research Fund of Guangdong Province under Grant no. 2015B010131001.

References

[1] T. Ma, J. Zhou, M. Tang et al., "Social network and tag sources based augmenting collaborative recommender system," *IEICE Transactions on Information and Systems*, vol. E98D, no. 4, pp. 902–910, 2015.

[2] S. Hu, Q. Wang, J. Wang, Z. Qin, and K. Ren, "Securing SIFT: privacy-preserving outsourcing computation of feature extractions over encrypted image data," *IEEE Transactions on Image Processing*, vol. 25, no. 7, pp. 3411–3425, 2016.

[3] L. Wang, K. Wu, J. Xiao, and M. Hamdi, "Harnessing frequency domain for cooperative sensing and multi-channel contention in CRAHNs," *IEEE Transactions on Wireless Communications*, vol. 13, no. 1, pp. 440–449, 2014.

[4] V. T. N. Nguyen and R. Kirner, "Throughput-driven partitioning of stream programs on heterogeneous distributed systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 3, pp. 913–926, 2016.

[5] S. Chen and X. Li, "Input-adaptive parallel sparse fast Fourier transform for stream processing," in *Proceedings of the 28th ACM International Conference on Supercomputing (ICS '14)*, pp. 93–102, ACM, June 2014.

[6] M.-H. Kuo, S.-L. Wang, and W.-T. Chen, "Using information and mobile technology improved elderly home care services," *Health Policy and Technology*, vol. 5, no. 2, pp. 131–142, 2016.

[7] Y. Kong, M. Zhang, and D. Ye, "A belief propagation-based method for task allocation in open and dynamic cloud environments," *Knowledge-Based Systems*, vol. 115, pp. 123–132, 2017.

[8] G. Wang, Y. Zou, Z. Zhou, K. Wu, and L. M. Ni, "We can hear you with Wi-Fi!," *IEEE Transactions on Mobile Computing*, vol. 15, no. 11, pp. 2907–2920, 2016.

[9] K. Wu, H. Li, L. Wang et al., "HJam: attachment transmission in WLANs," *IEEE Transactions on Mobile Computing*, vol. 12, no. 12, pp. 2334–2345, 2013.

[10] A. P. Oghuma, C. F. Libaque-Saenz, S. F. Wong, and Y. Chang, "An expectation-confirmation model of continuance intention to use mobile instant messaging," *Telematics and Informatics*, vol. 33, no. 1, pp. 34–47, 2016.

[11] A. P. Oghuma, Y. Chang, C. F. Libaque-Saenz, M.-C. Park, and J. J. Rho, "Benefit-confirmation model for post-adoption behavior of mobile instant messaging applications: a comparative analysis of KakaoTalk and Joyn in Korea," *Telecommunications Policy*, vol. 39, no. 8, pp. 658–677, 2015.

[12] S. O. Ogara, C. E. Koh, and V. R. Prybutok, "Investigating factors affecting social presence and user satisfaction with mobile instant messaging," *Computers in Human Behavior*, vol. 36, pp. 453–459, 2014.

[13] Y.-F. Kuo and S.-N. Yen, "Towards an understanding of the behavioral intention to use 3G mobile value-added services," *Computers in Human Behavior*, vol. 25, no. 1, pp. 103–110, 2009.

[14] S. Hong, J. Y. L. Thong, and K. Y. Tam, "Understanding continued information technology usage behavior: a comparison of three models in the context of mobile internet," *Decision Support Systems*, vol. 42, no. 3, pp. 1819–1834, 2006.

[15] H. Chen, H. Jin, and S. Wu, "Minimizing inter-server communications by exploiting self-similarity in online social networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 4, pp. 1116–1130, 2016.

[16] C. Sun, Y. Wang, J. Zheng, and D. F. Hsu, "Feature fusion for mobile usage prediction using rank-score characteristics," in *Proceedings of the 12th IEEE International Conference on Cognitive Informatics and Cognitive Computing, (ICCI&CC '13)*, pp. 212–217, July 2013.

[17] C. Shin, J.-H. Hong, and A. K. Dey, "Understanding and prediction of mobile application usage for smart phones," in

Proceedings of the 14th International Conference on Ubiquitous Computing (UbiComp '12), pp. 173–182, September 2012.

[18] Y. Ma, S. Wang, P. C. K. Hung, C.-H. Hsu, Q. Sun, and F. Yang, "A highly accurate prediction algorithm for unknown web service QoS values," *IEEE Transactions on Services Computing*, vol. 9, no. 4, pp. 511–523, 2016.

[19] Y. Zou, J. Xiao, J. Han, K. Wu, Y. Li, and L. M. Ni, "GRfid: a device-free RFID-based gesture recognition system," *IEEE Transactions on Mobile Computing*, vol. 16, no. 2, pp. 381–393, 2017.

[20] K. Zhu, X. Zhang, B. Xiang, and L. Zhang, "Exploiting user context and network information for mobile application usage prediction," in *Proceedings of the 7th International Workshop on Hot Topics in Planet-scale mObile computing and online Social neTworking (HOTPOST '15)*, pp. 25–30, Hangzhou, China, June 2015.

[21] Z.-X. Liao, P.-R. Lei, T.-J. Shen, S. C. Li, and W. C. Peng, "Appnow: predicting usages of mobile applications on smart phones," in *Proceedings of the Conference on Technologies and Applications of Artificial Intelligence (TAAI '12)*, pp. 300–303, IEEE, Tainan, Taiwan, November 2012.

[22] Z.-X. Liao, Y.-C. Pan, W.-C. Peng, and P.-R. Lei, "On mining mobile apps usage behavior for predicting apps usage in smartphones," in *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management (CIKM '13)*, pp. 609–618, ACM, November 2013.

[23] Y. Xu, "The application of ARIMA model in Chinese mobile user prediction," in *Proceedings of the IEEE International Conference on Granular Computing (GrC '10)*, pp. 586–591, August 2010.

[24] I. Kloumann, L. Adamic, J. Kleinberg, and S. Wu, "The lifecycles of apps in a social ecosystem," in *Proceedings of the 24th International Conference on World Wide Web (WWW '15)*, pp. 581–591, May 2015.

[25] Y. Zhang, X. Sun, and W. Baowei, "Efficient algorithm for k-barrier coverage based on integer linear programming," *China Communications*, vol. 13, no. 7, pp. 16–23, 2016.

[26] Y. Zheng, B. Jeon, D. Xu, Q. M. J. Wu, and H. Zhang, "Image segmentation by generalized hierarchical fuzzy C-means algorithm," *Journal of Intelligent and Fuzzy Systems*, vol. 28, no. 2, pp. 961–973, 2015.

[27] Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang, "Enabling personalized search over encrypted outsourced data with efficiency improvement," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 9, pp. 2546–2559, 2016.

[28] Q. Wang, M. He, M. Du, S. S. Chow, R. W. Lai, and Q. Zou, "Searchable encryption over feature-rich data," *IEEE Transactions on Dependable and Secure Computing*, 2016.

[29] Y. Wang, K. Wu, and L. M. Ni, "Wifall: device-free fall detection by wireless networks," *IEEE Transactions on Mobile Computing*, vol. 16, no. 2, pp. 581–594, 2017.

[30] M. R. Haddad, H. Baazaoui, D. Ziou, and H. Ben Ghezala, "A predictive model for recurrent consumption behavior: an application on phone calls," *Knowledge-Based Systems*, vol. 64, pp. 32–43, 2014.

[31] I. Valera and M. Gomez-Rodriguez, "Modeling adoption and usage of competing products," in *Proceedings of the IEEE International Conference on Data Mining Series (ICDM '15)*, pp. 409–418, Atlantic City, NJ, USA, 2015.

[32] F. Zhang, H. Chen, and H. Jin, "Piggyback game: efficient event stream dissemination in Online Social Network systems," in *Proceedings of the IEEE 24th International Conference on Network Protocols (ICNP '16)*, pp. 1–10, Singapore, November 2016.

[33] L. Wang, Z. Zhou, and W. Wu, "Game theory-based model for maximizing SSP utility in cognitive radio networks," *Computer Communications*, vol. 86, pp. 29–39, 2016.

[34] Q. Zhao, "Data acquisition and simulation of natural phenomena," *Science China Information Sciences*, vol. 54, no. 4, pp. 683–716, 2011.

[35] P. D. Taylor and L. B. Jonker, "Evolutionarily stable strategies and game dynamics," *Mathematical Biosciences*, vol. 40, no. 1-2, pp. 145–156, 1978.

[36] S. Goyal, H. Heidari, and M. Kearns, "Competitive contagion in networks," *Games and Economic Behavior*, 2014.

[37] B. Gu and V. S. Sheng, "A robust regularization path algorithm for ν-support vector classification," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–8, 2016.

[38] X. Wen, L. Shao, Y. Xue, and W. Fang, "A rapid learning algorithm for vehicle classification," *Information Sciences*, vol. 295, pp. 395–406, 2015.

[39] Z. Fu, X. Sun, S. Ji, and G. Xie, "Towards efficient content-aware search over encrypted outsourced data in cloud," in *Proceedings of the 35th Annual IEEE International Conference on Computer Communications (INFOCOM '16)*, pp. 1–9, IEEE, San Francisco, Calif, USA, April 2016.

[40] Y. Chen, C. Hao, W. Wu, and E. Wu, "Robust dense reconstruction by range merging based on confidence estimation," *Science China Information Sciences*, vol. 59, no. 9, Article ID 092103, 11 pages, 2016.

[41] H. Li, K. Wu, Q. Zhang, and L. M. Ni, "CUTS: improving channel utilization in both time and spatial domain in WLANs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 6, pp. 1413–1423, 2014.

Hindawi
Scientific Programming
Volume 2017, Article ID 5081526, 10 pages
https://doi.org/10.1155/2017/5081526

Research Article

Parallelizing Gene Expression Programming Algorithm in Enabling Large-Scale Classification

Lixiong Xu, Yuan Huang, Xiaodong Shen, and Yang Liu

School of Electrical Engineering and Information, Sichuan University, Chengdu 610065, China

Correspondence should be addressed to Yang Liu; yang.liu@scu.edu.cn

Received 19 October 2016; Accepted 23 January 2017; Published 20 February 2017

Academic Editor: Alex M. Kuo

As one of the most effective function mining algorithms, Gene Expression Programming (GEP) algorithm has been widely used in classification, pattern recognition, prediction, and other research fields. Based on the self-evolution, GEP is able to mine an optimal function for dealing with further complicated tasks. However, in big data researches, GEP encounters low efficiency issue due to its long time mining processes. To improve the efficiency of GEP in big data researches especially for processing large-scale classification tasks, this paper presents a parallelized GEP algorithm using MapReduce computing model. The experimental results show that the presented algorithm is scalable and efficient for processing large-scale classification tasks.

1. Introduction

In recent years, Gene Expression Programming (GEP) [1] algorithm has been widely studied due to its significant function mining ability. Comparing to the other machine learning algorithms such as support vector machine and neural networks, the most remarkable characteristic of GEP is that it can explicitly mine the mathematical equation f of the dependent variable $\mathbf{y}$ and independent variables $\mathbf{x}$ from dataset. As a result, the equation $\mathbf{y} = f(\mathbf{x})$ can be easily stored and employed in future study of the data. Similar to the Genetic Algorithm, GEP algorithm also simulates the processes of biological evolution to mine a function with the best fitness to represent the data relations. During the evolution, the algorithm employs selection, crossover, and mutation operations to generate offspring. Each individual of the offspring is assessed by a fitness function. The individual that has a better fitness has a higher chance to be selected to produce a next generation. The evolution keeps evolving until a satisfied function that can describe the data relations is found.

As an effective data analyzing approach, classification has been researched a lot. The classification algorithms, especially supervised classification algorithms, for example, artificial neural networks (ANNs), show remarkable classification abilities. However, ANNs are also function-fitting algorithms fundamentally, although the algorithms cannot output the functions explicitly as GEP does. This point motivates us that GEP can also be employed to deal with the supervised classification tasks using the following idea:

(1) Let the training data be $\mathbf{x}$ and the encoded classes be $\mathbf{y}$.

(2) Train the GEP algorithm using $\mathbf{x}$ and $\mathbf{y}$ to mine a function f.

(3) Input the to-be-classified data $\mathbf{x_t}$ to f; observe the output $\mathbf{y_c}$, which can represent the classes that $\mathbf{x_t}$ should belong to. Therefore, $\mathbf{x_t}$ is classified.

It also motivates us that, based on the works [2, 3], GEP-based classification has great potential to be further applied into large-scale classification tasks.

Unfortunately several works [4–6] pointed out that to process large-scale tasks using GEP may encounter the low efficiency issue. The reason is that, as a heuristic algorithm, GEP needs an extremely long time to mine the best-fitted function for large volume of data. Therefore to improve the large-scale classification efficiency using GEP is also focused by this paper. As a result, this paper presents a parallelized GEP algorithm in enabling large-scale classification. The algorithm is designed and implemented in the MapReduce

distributed computing environment. Following a number of tests based on standard benchmark datasets have been carried out. The experimental results reveal that the parallelized GEP algorithm shows advantages in dealing with large-scale classification tasks.

The rest of the paper is organized as follows: Section 2 reviews the related work; Section 3 presents the parallelization of GEP; Section 4 discusses the experimental results; and Section 5 concludes the paper.

2. Related Work

As an effective function mining algorithm, GEP has been widely applied in numbers of researches. Sabar et al. [7] employed GEP to design a hyperheuristic framework in order to solve the combinatorial optimization problems. Their experimental results show that the proposed framework has great potential to solve the problems. Hwang et al. [8] employed GEP to predict the Qos (Quality of Service) traffic in the Ethernet passive optical network. The authors combined GEP algorithm to tackle the queue variation during waiting times as well as reducing the high priority packet delay. Deng et al. [9] also employed GEP and rough set to assess the security risk in cyber physical power system. Based on their studies, security risk levels of cyber physical power system can be accurately predicted.

However, several works [4–6] pointed out that GEP has low efficiency issue for processing complicated tasks. To solve the issue, several researchers focused on improving the algorithm parameters of GEP. Xue and Wu [10] proposed Symbiotic Gene Expression Programming (SGEP) based on symbiotic algorithm, estimation of distribution algorithm, and evolution processes improved GEP. The experimental results indicate that SGEP outperforms GEP in terms of efficiency. Chen et al. [11] pointed out that the most computationally expensive computation of GEP is the evolution in the expression tree. Therefore they proposed Reduced-GEP algorithm which is based on the chromosome reduction. The experimental results show that the algorithm is effective and efficient in calculating the fitness and reducing the size of chromosome. In research [12], inspired by the diversity of chromosome arrangements in biology, an unconstrained encoded Gene Expression Programming was proposed. The approach can enlarge the function searching space, which enhances the parallelism and the adaptability of the standard GEP algorithm.

Another effective way of solving the efficiency issue of GEP is to use parallel computing or distributed computing. Du et al. [13] proposed an asynchronous distributed parallel GEP algorithm. They aimed at speeding up the convergence of finding the optimal solution using MPI (Message Passing Interface) [14]. In each processor, a standalone GEP algorithm is running. And then the processors exchange their best individuals and continue evolving. Until a termination message is sent to the processors, the algorithm stops. Based on the experimental results, the authors claimed that the proposed algorithm can greatly speed up the algorithm convergence. However, they have not evaluated the algorithm using large volume of data. And also MPI is highly depending on the homogeneous hardware environment, which limits the algorithm adaption. Du et al. also proposed a MapReduce [15] based distributed GEP algorithm to process large populations and datasets. Similar to [13], each Map computes the fitness and in each Reduce the selection, mutation, and crossover operations are executed. The output is output into the distributed file system where the exchanges of the best individuals occur. Although the authors claimed that they achieved algorithm speedup, two issues should be discussed. Firstly the algorithm needs a large number of Reduces, which generates system overhead because of IO operations [2] in reducers. Secondly their algorithm needs a large number of iterations. However, MapReduce does not support iteration originally. Instead, the algorithm has to submit a number of MapReduce jobs to the cluster, which generates tremendously large overhead [3]. Browne and dos Santos [16] also discussed the parallelization of GEP using the island model. However, their algorithm does not focus on the parallelization. And the algorithm performance for processing the large volume of dataset has not been evaluated.

The improvement of GEP presented in this paper mainly focuses on parallelizing the GEP algorithm for executing large-scale classification. Our algorithm first employs the Hadoop framework [17] as the underlying infrastructure. And secondly combining with the ensemble techniques, the algorithm is able to supply efficiency, scalability, and accuracy.

3. Algorithm Design

3.1. Classification Using GEP. Based on the selection, crossover, mutation, and fitness, GEP is able to mine a function from the given dataset. Therefore, let T denote the training dataset; t_i denote an instance in T; l denote the length of t_i; j denote the jth class of T; T_e denote the testing dataset; t_{e_i} denote an instance in T_e; c_j denote the coded identifier of class; σ denote a threshold; r represent the number of correctly classified instances; and n represent the number of training instances. The classification using GEP is shown in Algorithm 1.

3.2. MapReduce and Hadoop. MapReduce computing model contributes two main functions Map and Reduce to facilitate the development of the distributed computing applications. Map function executes main computation and Reduce function collects the intermediate output of Maps and generates the final output. Each Map computes the data instances one by one in the form of key-value pair $\{K1, \text{Value1}\}$. And then the computed result is output as an intermediate output $\{K2, \text{Value2}\}$. The Reduces collects the intermediate output of all Maps. Afterwards each Reduce merges the inputs having the same key and generates the final output.

Hadoop framework [17, 18] is a Java based implementation of MapReduce. Two types of nodes including one NameNode and several DataNodes consist of a typical Hadoop cluster. The NameNode manages the metadata, whilst the DataNode executes a number of Map (mappers) and Reduce (reducers) operations in parallel. Both the NameNode and DataNodes contribute their resources including processors, memory, hard disks, and network adaptors to form the Hadoop

(1) For each $t_i \in T$
  Encode c_j representing the class of t_i
(2) For each t_i in T
    Input t_i and c_j into GEP
(3) Let c_j be $\mathbf{y}$ and t_i be $\mathbf{x}$.
(4) Initiate GEP components: function set, link function, selection
  mutation, crossover and fitness r/n.
(5) In each generation, GEP mines $c = f(t_i)$,
  if $|c - c_j| \leq \sigma$
    $r = r + 1$
(6) GEP keeps running
  until terminating condition (determined generation or fitness) met
(7) Output f which represents $c_j = f(t_i)$
(8) Let σ denote a threshold
  For each t_{e_i} in T_e
    Compute $y_e = f(t_{e_i})$
    if $|y_e - c_j| \leq \sigma$
      $t_{e_i} \in c_j$
    else
      t_{e_i} is an outlier
(9) Classification terminates

Algorithm 1: The pseudocode of classification using GEP.

Distributed File System (HDFS) [17]. HDFS not only is responsible for high performance data storage but also manages data processing for the mappers and reducers, which supplies fault tolerant, load balancing, scalability, and heterogeneous hardware support. Figure 1 shows the structure of Hadoop framework.

3.3. Parallelization of GEP in Enabling Large-Scale Classification. In the training phase, let m denote the number of mappers. Therefore, the training dataset T can be divided into a number of m data chunks, in which each chunk is represented by T_i $i \in m$. T_i satisfies

$$\bigcup_{i=1}^{m} T_i = T. \tag{1}$$

Firstly m mappers input m data chunks in parallel. And then each mapper initiates one sub-GEP starting the function mining according to the input training data. As long as a function (a classifier) has been mined in each mapper, totally a number of m classifiers are generated.

In the classification phase, the testing dataset T_e is also divided into m chunks. Each previously trained classifier inputs one testing data chunk and executes the classification. Therefore, the classification using GEP can be parallelized. However, one problem should be mentioned that, due to the data separation of the training dataset, each sub-GEP in each mapper is only trained by a subset of the original dataset. The insufficient training may lead to the loss of classification accuracy. In order to parallelize GEP classification avoiding accuracy loss, ensemble techniques including bootstrapping and majority voting are employed.

Bootstrapping is based on the idea of controlling the number of times that the training instances appear in the bootstrapping samples, so that, in the number of B bootstrapping samples, each instance appears the same number of times [19]. To create balanced bootstrapping samples, the following steps could be followed:

(1) Construct a string of the instances $X_1, X_2, X_3, \ldots, X_n$ repeating B times so that a sequence $Y_1, Y_2, Y_3, \ldots, Y_{Bn}$ is achieved.

(2) A random permutation p of the integers from 1 to B_n is taken. Therefore the first bootstrapping sample can be created from $Y_p(1), Y_p(2), Y_p(3), \ldots, Y_p(n)$, moreover the second bootstrapping sample from $Y_p(n + 1), Y_p(n + 2), Y_p(n + 3), \ldots, Y_p(2n)$, and so on.

(3) Repeat step (2) until $Y_p((B - 1)n + 1), Y_p((B - 1)n + 2), Y_p((B - 1)n + 3), \ldots, Y_p(Bn)$, which is the Bth bootstrapping sample.

Based on the bootstrapping, the instance distributions of the original dataset can be simulated in the samples, in which the original data information can be kept more. The majority voting is a commonly used combination technique. The ensemble classifier predicts a class for a test instance that is predicted by the majority of the base classifiers [20].

By employing the bootstrapping and majority voting, the parallelized GEP classification algorithm works as follows.

In the training phase:

(1) The algorithm firstly generates a number of m bootstrapped sample sets using the training dataset T. Each set T_i is saved in one data chunk stored in the HDFS.

(2) Each mapper initiates a sub-GEP and inputs one data chunk from HDFS.

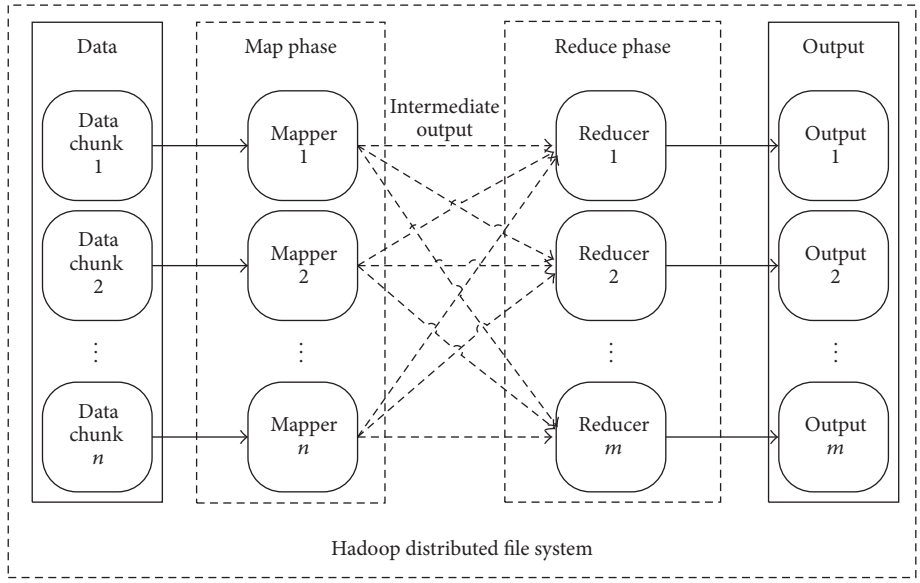


Figure 1: The structure of Hadoop framework.

(3) Each mapper trains its sub-GEP according to step (1) to (6) in Algorithm 1. As long as the training terminates, the mined function f_i is collected by reducer and saved into HDFS in the value pairs $\langle mapperId, f_i \rangle$.

(4) As in each mapper its sub-GEP mines the function individually, therefore, finally a number of m different functions can be saved, which means a number of m weak classifiers are created.

Figure 2 shows the training phase of the algorithm.

In the classification phase:

(1) Each mapper retrieves one function f_i from HDFS so that the mapper becomes one weak classifier.

(2) And then all the mappers input the same one instance t_{e_i} from the testing dataset T_e.

(3) In each mapper, when t_{e_i} is input, a value of c can be computed according to f_i. Comparing the values of c_j, σ, and c according to step (8) in Algorithm 1, t_{e_i} can be classified.

(4) The gth mapper outputs its intermediate output $\langle t_{e_i}, result_g \rangle$.

(5) One reducer collects all the intermediate outputs from all the m mappers. And then, it merges the outputs having the same key into one group. In the group, the majority voting is executed to vote the final classification result.

Figure 3 shows the classification phase.

4. Algorithm Evaluation

4.1. Experimental Environment. In order to evaluate the algorithm performance, a physical Hadoop cluster constituted by one NameNode and four DataNodes is established. The details of the cluster are listed in Table 1.

Table 1: The cluster details.

NameNode	CPU: Core i7@3 GHz Memory: 8 GB SSD: 750 GB OS: Fedora
DataNodes	CPU: Core i7@3.8 GHz Memory: 32 GB SSD: 250 GB OS: Fedora.
Network bandwidth	1 Gbps
Hadoop version	2.3.0

Table 2: The dataset details.

Type	Iris	Wine
Dataset characteristics	Multivariate	Multivariate
Instance number	150	178
Attribute number	4	13
Class number	3	3

The datasets employed in the experiments are Iris dataset [21] and Wine dataset [22]. The details of the datasets are listed in Table 2.

The parameters of GEP used in the experiments are listed in Table 3.

4.2. Accuracy of the Classification. Let *rightNum* represent the number of the correctly classified instances and *wrongNum*

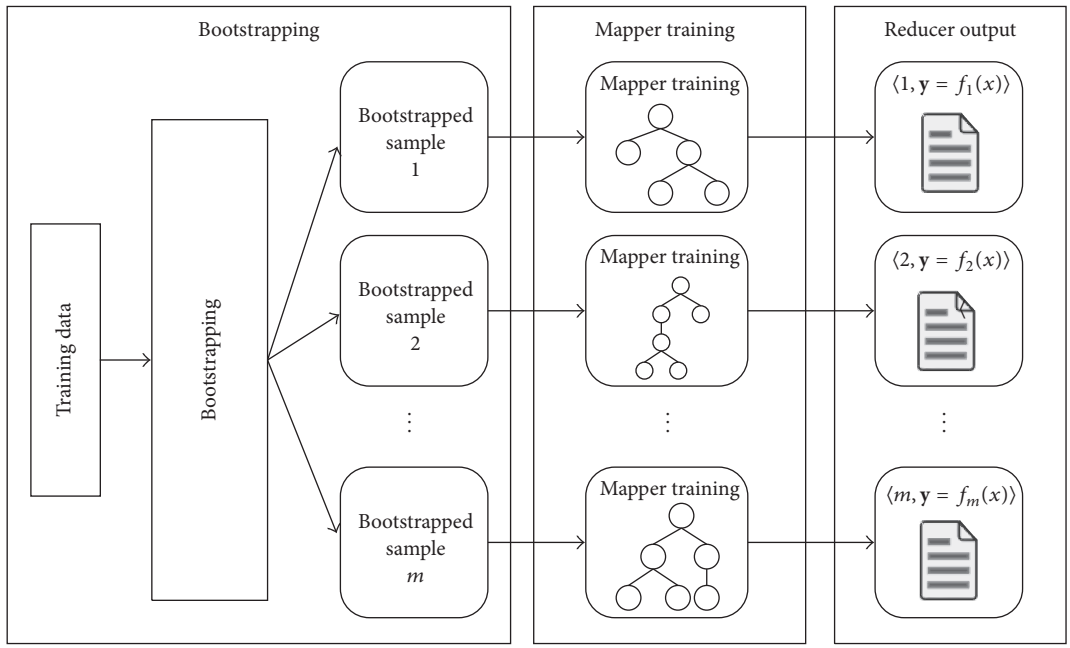


Figure 2: Training phase in the classification.

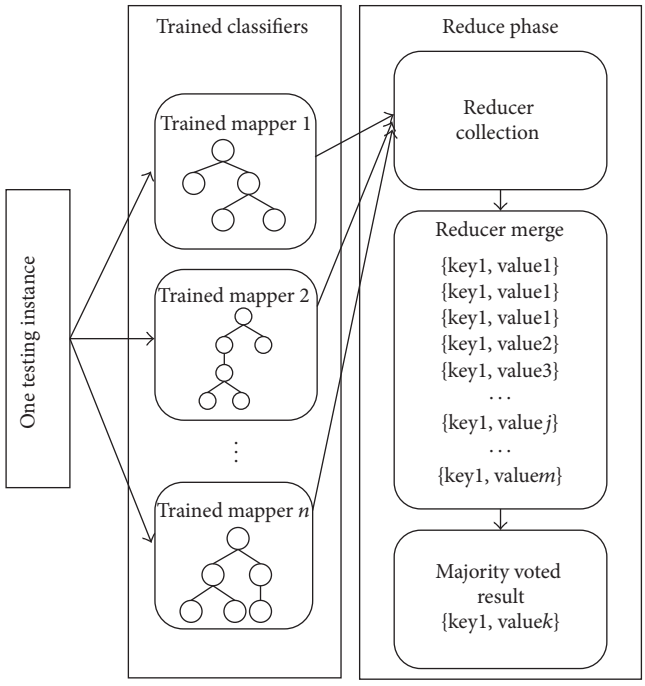


Figure 3: The classification of the presented algorithm.

represent the number of the wrongly classified instances. Therefore the classification accuracy p is defined as

$$p = \frac{rightNum}{rightNum + wrongNum} \times 100\%. \quad (2)$$

In the following tests, increasing numbers of instances are selected from the two datasets as the training instances, whilst the rest instances are the testing instances. The bootstrapping number is four and the algorithm starts eleven mappers for executing the parallelized GEP classification. The experimental results are shown in Figures 4–14 and *The Functions in Each Sub-GEP for Classifying Iris Dataset* and *The Functions in Each Sub-GEP for Classifying Wine Dataset* in the appendix.

Figure 4 shows the classification accuracy of the Iris dataset with an increasing number of the training instances from 12 to 105. It can be observed that the parallel GEP algorithm performs highly stable and outperforms the standalone GEP algorithm. The visualizations of the classification results

Table 3: The details of the parameters of GEP.

Number of genes	2 (iris)/4 (wine)
Linking function	+
Head length	6
Function set	+ − ∗/cos sin tan exp log sqrt abs
Fitness	r/n
Terminal set	*abcd* (iris)/*abcdefghijklm* (wine)
Population size	100
Mutation rate	0.044
IS transposition rate	0.1
RIS transposition rate	0.1
Gene transposition rate	0.1
One-point recombination rate	0.4
Two-point recombination rate	0.2
Gene recombination rate	0.1
Threshold σ	0.5
Coded classes	1, 2, and 3

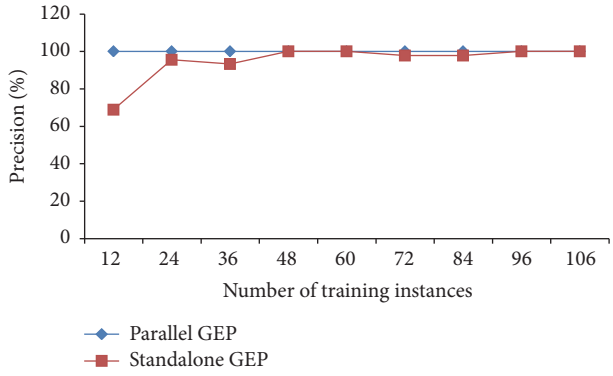


Figure 4: The precision comparison for classifying Iris dataset.

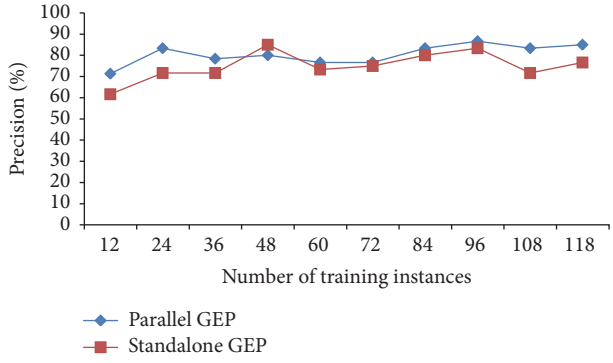


Figure 5: The precision comparison for classifying Wine dataset.

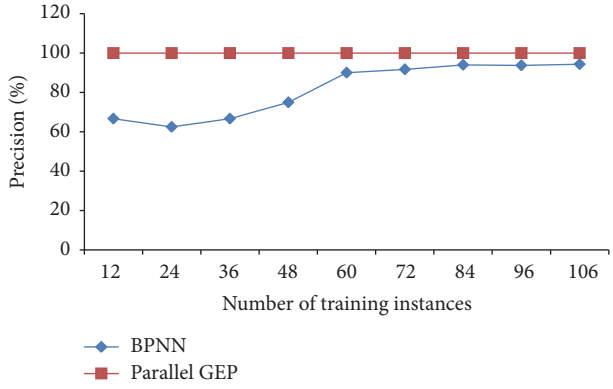


Figure 6: The precision comparison for classifying Iris dataset using parallel GEP and BPNN.

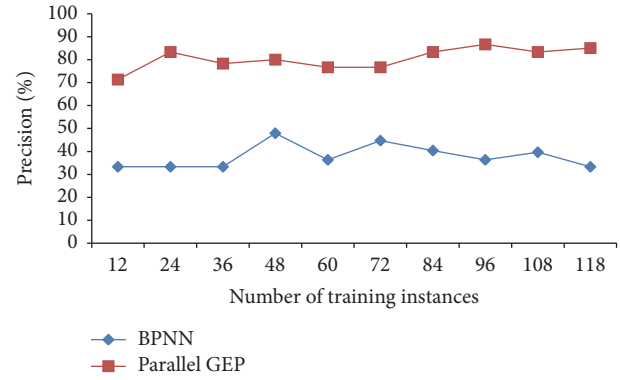


Figure 7: The precision comparison for classifying Wine dataset using parallel GEP and BPNN.

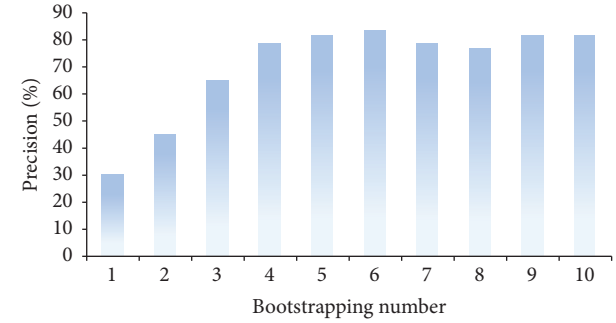


Figure 8: Precision of the classification with increasing bootstrapping numbers.

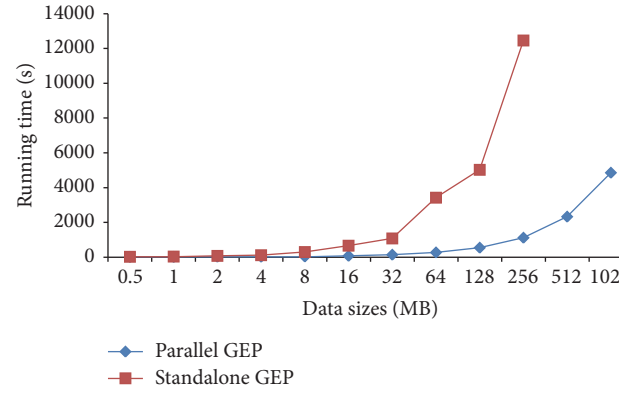


Figure 9: Algorithm efficiency comparison of increasing training data sizes.

are shown by Figures 11, 12, and *The Functions in Each Sub-GEP for Classifying Iris Dataset* in the appendix.

Wine dataset has also been employed to evaluate the classification accuracy. Comparing to Iris dataset, each instance of Wine dataset has 13 attributes, which may impact the classification accuracy. The experimental result is shown in Figure 5.

Figure 5 shows the classification accuracy of the Wine dataset with an increasing number of the training instances from 12 to 118. The result shows that, due to more attributions of the instances, the accuracy of the parallel GEP starts fluctuating. However, in most of the tests the parallel GEP still outperforms the standalone GEP in terms of accuracy. It further indicates the ensemble techniques can help to improve the classification accuracy. The visualizations of the classification results are shown by Figures 13, 14, and *The Functions in Each Sub-GEP for Classifying Wine Dataset* in the appendix.

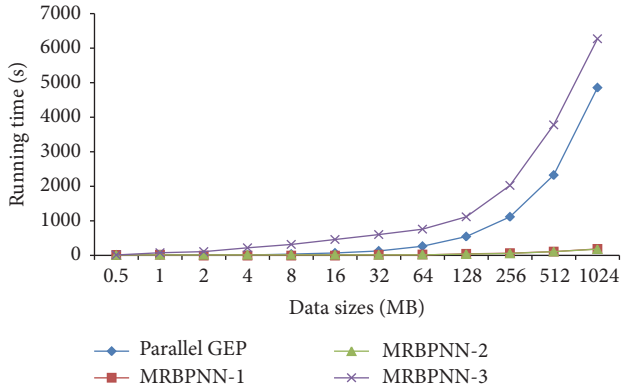


Figure 10: Comparison of the processing time of the parallel BPNNs and the parallel GEP with increasing training data sizes.

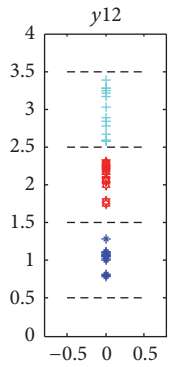


Figure 11: The classification result of the standalone GEP using Iris dataset.

For further evaluating the effectiveness of the proposed algorithm, we also implemented backpropagation neural network (BPNN). The comparisons of the classification accuracy are shown in Figure 6.

Figure 6 indicates that, in terms of Iris dataset classification accuracy, the parallel GEP algorithm outperforms BPNN. Although the neural network also performs well in the classification, when the number of the training instances is small, it gives lower classification accuracy.

Figure 7 indicates that, in terms of Wine dataset classification accuracy, the parallel GEP algorithm greatly outperforms BPNN. Due to more attributions in the dataset, it is difficult for BPNN to correctly classify most of the testing instances. Contrarily, parallel GEP can still keep a higher accuracy.

It should be noticed that the bootstrapping number which represents the number of times the training instances appear in the bootstrapping samples also impacts the algorithm accuracy. Therefore Figure 8 is generated to show the classification results with increasing bootstrapping numbers. Wine dataset is selected as the experimental dataset, in which a number of 118 instances are the training instances and the remaining 60 instances are the testing instances.

In Figure 8, it can be observed that when the bootstrapping number is less than 6, the classification precision keeps increasing. And then, the precision varies slightly. Figure 8 significantly tells that initially enlarging the bootstrapping number improves the classification. However, when the bootstrapping number reaches a certain value, the performance in terms of classification precision cannot be further improved.

4.3. Running Time of the Classification. In this section, Wine dataset is selected as the experimental dataset. The algorithm processing time for the increasing training data sizes has been evaluated. In the following tests, firstly the bootstrapping number is 4, which means each training instance appears 4 times. The number of the training instances is 118 whilst the testing instances remain 60. And then the training data size is duplicated from approximately 0.5 MB to 1024 MB. It should be pointed out that, because of the duplication, the bootstrapping number will change from 4 to $4n$, where n represents the duplicated times. However, this section only focuses on the algorithm efficiency. Therefore although the varying bootstrapping numbers may affect the classification precision slightly according to Figure 8, the algorithm processing time with increasing training data sizes is highlighted in Figure 9.

Figure 9 shows that when the training data size is small, the performances of the standalone GEP and the parallel GEP are nearly the same. However, when the data size becomes larger, the parallel GEP outperforms the standalone GEP. When the data size increases more than 256 MB, the standalone GEP cannot finish the classification due to memory limitation. Contrarily, the parallel GEP still works fine even if the data size increases to 1024 MB.

To further compare the classification efficiency to the other classification algorithms, the MapReduce based parallel bac propagation neural network algorithms (MRBPNN 1, 2, and 3) [2] are also implemented. The comparisons are shown in Figure 10.

Figure 10 shows that in terms of the algorithm running time, the parallel BPNN algorithms MRBPNN 1 and 2 outperform the parallel GEP. The main reason is that GEP needs longer time to evolve. Contrarily, MRBPNN 1 and 2 need shorter time to train the neurons. Although parallel GEP performs slower than MRBPNN 1 and 2 do, it can supply higher classification accuracy according to Figures 6 and 7.

5. Conclusion

This paper presents a MapReduce and ensemble techniques based parallel Gene Expression Programming algorithm in enabling large-scale classification. The parallelization of GEP mainly focuses on paralleling the training phase (function mining phase) which is the most time consuming and computational intensive process. The experimental results show that the presented algorithm outperforms the standalone GEP and BPNN in terms classification accuracy. In the algorithm executing time evaluations, the presented parallel GEP also shows remarkable performance comparing to the standalone GEP. Although the parallel GEP works slower than MRBPNN 1 and 2, it can supply higher classification accuracy, which enables the presented parallel GEP to be one of the effective tools dealing with large-scale classification.

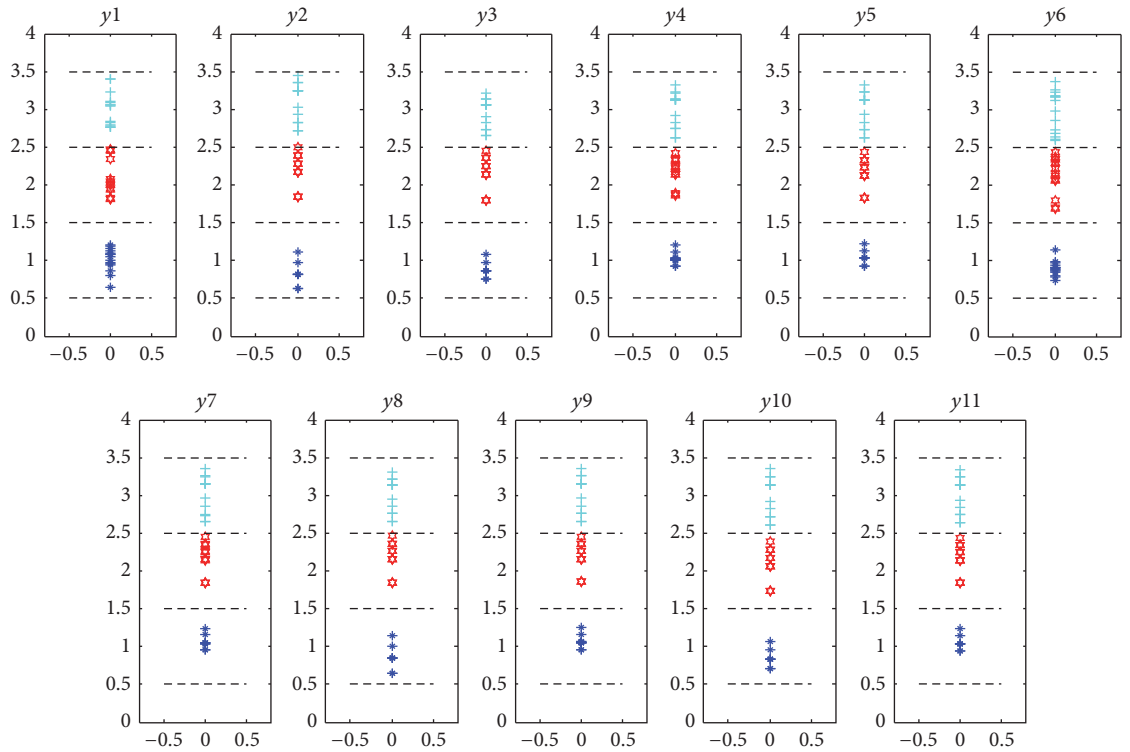


Figure 12: The classification results of eleven sub-GEPs.

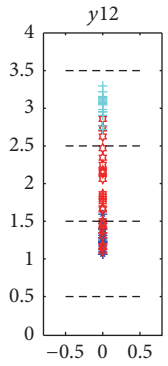


Figure 13: The classification result of the standalone GEP using Wine dataset.

Appendix

In the appendix, the details of classifying Iris and Wine dataset have been listed. Figure 11 visualizes the classification result of the standalone GEP for classifying Iris dataset.

Figure 11 indicates that GEP has ability to mine a function f to classify instances into three classes. In this case, the mined function f is represented by $y12$:

$$y12 = (d) + (\cos(\sin(\cos(\text{abs}(((b) * (d)) * (c)))))) \quad \text{(A.1)}$$

The Iris dataset classification results of the eleven sub-GEPs employed by parallel GEP are shown in Figure 12.

Figure 12 shows that the eleven sub-GEPs have different classification results due to differently mined functions f. However, because of the majority voting in the reducer, parallel GEP is able to output a correct classification result. The eleven mined functions are listed as follows.

The Functions in Each Sub-GEP for Classifying Iris Dataset

$$y1 = (d) + (\cos(\sin((\tan(\exp(d))) * ((a)/(c)))));$$

$$y2 = (d) + (\sin(\text{sqrt}(\text{sqrt}(d))));$$

$$y3 = (\cos(\cos(\text{abs}(\cos(\cos(-(d))))))) + (d);$$

$$y4 = (\cos(\sin((\log(b))/(\text{sqrt}(b))))) + (d);$$

$$y5 = (d) + (\cos(\sin(\exp(\tan(\exp((b)/(b))))))));$$

$$y6 = (d) + (\text{abs}(\cos(-(-(\sin(\text{sqrt}(a))))))));$$

$$y7 = (-(-(d))) + (\cos((\cos(\sin(b))) * (\cos(\cos(b)))));$$

$$y8 = (\text{sqrt}(\sin(\sin(\text{sqrt}(\sin(d)))))) + (d);$$

$$y9 = (\text{abs}(d)) + (\cos(\cos(\exp(\exp((-(a)) - (c))))));$$

$$y10 = (d) + (\cos(\text{abs}(\text{sqrt}(\cos(\text{sqrt}(\text{sqrt}(d)))))));$$

$$y11 = (\text{abs}(\sin(\exp((d) + (-(d)))))) + (d);$$

Figure 13 visualizes the classification result of the standalone GEP for classifying Wine dataset.

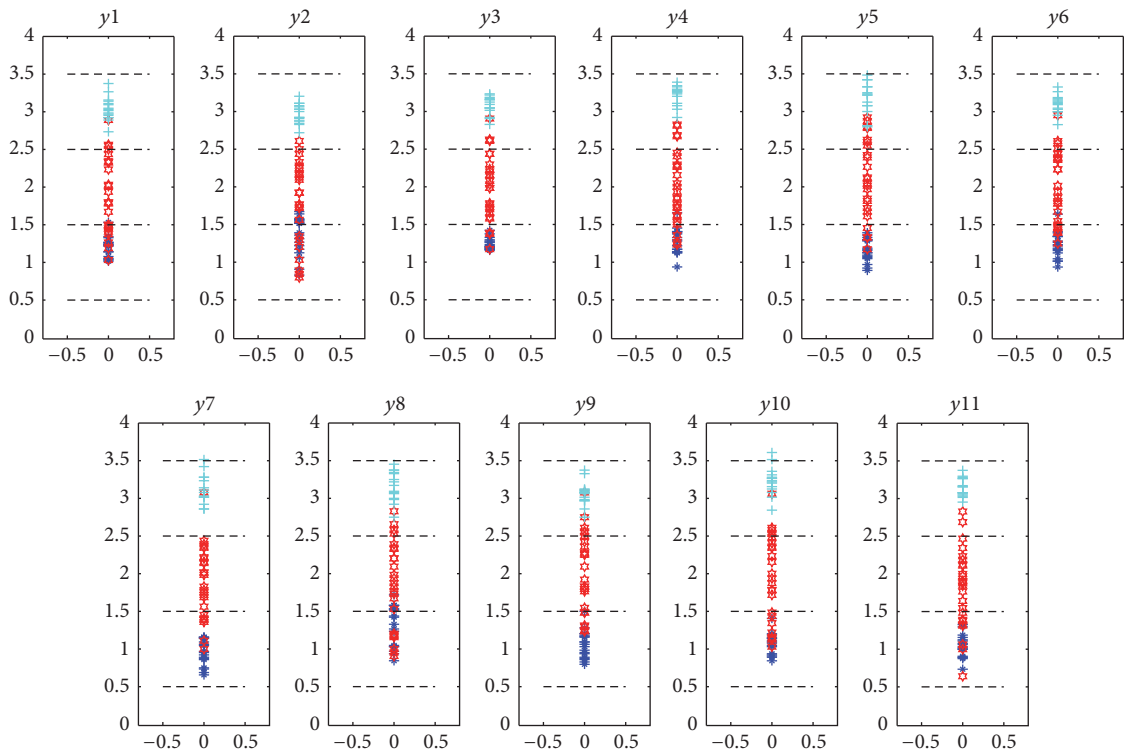


Figure 14: The classification results of eleven sub-GEPs.

In this case, the mined function f is represented by $y12$:

$$y12 = \left(\left(\left(\cos\left(\cos\left(\exp\left(\cos\left(\frac{(l)}{(g)}\right)\right)\right)\right)\right)\right) + \left(\cos\left(\cos\left(\cos\left((\cos(k)) - \left(\frac{(j)}{(g)}\right)\right)\right)\right)\right) + (\cos(g))\right) + (\cos(\cos(\exp(k)))) \quad \text{(A.2)}$$

The Wine dataset classification results of the eleven sub-GEPs employed by parallel GEP are shown in Figure 14.

The eleven mined functions are listed as follows.

The Functions in Each Sub-GEP for Classifying Wine Dataset

$y1 = (((\cos(\cos(\sqrt{(\sqrt{(m)} + (b)}))) + (\cos(k))) + (\sin(\sin(\cos(g))))) + (\sqrt{\sin(\text{abs}(\cos(\sin(\tan(k)))))})$

$y2 = (((\cos(\cos(\sqrt{\cos(h)}))) + ((\cos(\cos(j))) * ((\cos(h)) * (\cos(g))))) + (\text{abs}(\sin(\exp(k)))) + (\cos(\cos(\cos(\cos(\cos(\cos(g)))))))$

$y3 = (((\cos(\sin(\cos(\sin(-((a) + (g))))))) + (\sin(\cos(\sin(a))))) + (\sin(\sin(\cos(g))))) + (\cos(\cos((\cos(\text{abs}(g))) + (\sin(a)))))$

$y4 = (((\cos(\cos(\log(((c) * (l)) * (m))))) + (\cos(g))) + (\cos(\sin(f)))) + (\cos(\cos(\exp(k))))$

$y5 = (((\cos(h)) + (-(\sin(-((\sqrt{(d)}) * ((e)/(m))))))) + (\cos(g))) + (\cos(\cos((\cos(j))/k))))$

$y6 = (((\cos(\text{abs}(\cos(\exp(k))))) + (\cos(\cos(\cos(\cos(\sin(\log(d))))))))) + (\sin(\sin(\cos(g)))) + (\cos(\log((m) - (((k) - (e)) - (e)))))$

$y7 = (((\cos(\sqrt{(k)})) + (\cos(\sin(\text{abs}((\log(k)) + ((g) - (a))))))) + (\cos(\sin(\text{abs}((\log(m)) + (h)))))) + (\cos(g))$

$y8 = (((\cos(\sin(\sin((j) - ((l)/(a))))))) + (\text{abs}(\sin(\exp(\text{abs}(k)))))) + (\cos(\sin(\sin(\cos((c) + (g))))))) + (\sin(\cos(g)))$

$y9 = (((\text{abs}(\cos(\cos(\log(((m) - (d))/(f))))))) + (\cos(\cos(\exp(k))))) + (\cos(\tan(-(\tan(-(\sqrt{(a)})))))) + (\cos(g))$

$y10 = (((\cos(\sin(\cos(\sin(\cos(\sin(i))))))) + (\cos(g))) + (\cos(\sin(\log(((m) + (m)) + ((j) + (j))))))) + (\tan(\sin(\cos(\cos(\exp(k))))))$

$y11 = (((\cos(k)) + (\cos(g))) + (\sin(\exp(\cos((\cos(g)) + (\sin(a))))))) + (\sin(\sqrt{(g)}))$

Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this article.

Acknowledgments

The authors would like to appreciate the support from National Natural Science Foundation of China (no. 51437003).

References

[1] C. Ferreira, "Gene expression programming in problem solving," in *Soft Computing and Industry*, pp. 635–653, Springer, London, UK, 2002.

[2] Y. Liu, J. Yang, Y. Huang, L. Xu, S. Li, and M. Qi, "MapReduce based parallel neural networks in enabling large scale machine learning," *Computational Intelligence and Neuroscience*, vol. 2015, Article ID 297672, 13 pages, 2015.

[3] Y. Liu, L. Xu, and M. Li, "The Parallelization of back propagation neural network in MapReduce and Spark," *International Journal of Parallel Programming*, pp. 1–20, 2016.

[4] C. Zhou, W. Xiao, T. M. Tirpak, and P. C. Nelson, "Evolving accurate and compact classification rules with gene expression programming," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 6, pp. 519–531, 2003.

[5] S. Dehuri and S.-B. Cho, "Multi-objective classification rule mining using gene expression programming," in *Proceedings of the 3rd International Conference on Convergence and Hybrid Information Technology (ICCIT '08)*, pp. 754–760, November 2008.

[6] J. Wu, T. Li, B. Fang, Y. Jiang, Z. Li, and Y. Liu, "Parallel niche gene expression programming based on general multi-core processor," in *Proceedings of the International Conference on Artificial Intelligence and Computational Intelligence (AICI '10)*, pp. 75–79, October 2010.

[7] N. R. Sabar, M. Ayob, G. Kendall, and R. Qu, "Automatic design of a hyper-heuristic framework with gene expression programming for combinatorial optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 3, pp. 309–325, 2015.

[8] I.-S. Hwang, J.-Y. Lee, and A.-T. Liem, "Genetic expression programming: a new approach for QoS traffic prediction in EPONs," in *Proceedings of the 4th International Conference on Ubiquitous and Future Networks (ICUFN '12)*, pp. 249–254, July 2012.

[9] S. Deng, D. Yue, X. Fu, and A. Zhou, "Security risk assessment of cyber physical power system based on rough set and gene expression programming," *IEEE/CAA Journal of Automatica Sinica*, vol. 2, no. 4, pp. 431–439, 2015.

[10] S. Xue and J. Wu, "Gene expression programming based on symbiotic evolutionary algorithm," in *Proceedings of the 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC '11)*, pp. 3055–3058, August 2011.

[11] Y. Chen, C. J. Tang, R. Li, M. F. Zhu, C. Li, and J. Zuo, "Reduced-GEP: improving gene expression programming by gene reduction," in *Proceedings of the 2nd International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC '10)*, pp. 176–179, IEEE, Nanjing, China, August 2010.

[12] J. Zhang, Z. Wu, Z. Wang, J. Guo, and Z. Huang, "Unconstrained gene expression programming," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '09)*, pp. 2043–2048, May 2009.

[13] X. Du, L. Ding, and L. Jia, "Asynchronous distributed parallel gene expression programming based on estimation of distribution algorithm," in *Proceedings of the 4th International Conference on Natural Computation (ICNC '08)*, pp. 433–437, October 2008.

[14] Message Passing Interface, http://www.mcs.anl.gov/research/projects/mpi/.

[15] X. Du, Y. Ni, Z. Yao, R. Xiao, and D. Xie, "High performance parallel evolutionary algorithm model based on MapReduce framework," *International Journal of Computer Applications in Technology*, vol. 46, no. 3, pp. 290–295, 2013.

[16] N. P. A. Browne and M. V. dos Santos, "Adaptive representations for improving evolvability, parameter control, and parallelization of gene expression programming," *Applied Computational Intelligence and Soft Computing*, vol. 2010, Article ID 409045, 19 pages, 2010.

[17] Apache Hadoop, http://hadoop.apache.org.

[18] J. Venner, *Pro Hadoop*, Springer, New York, NY, USA, 2009.

[19] N. K. Alham, *Parallelizing support vector machines for scalable image annotation [Ph.D. thesis]*, Brunel University, Uxbridge, UK, 2011.

[20] N. K. Alham, M. Li, Y. Liu, and M. Qi, "A MapReduce-based distributed SVM ensemble for scalable image classification and annotation," *Computers and Mathematics with Applications*, vol. 66, no. 10, pp. 1920–1934, 2013.

[21] The Iris Dataset, https://archive.ics.uci.edu/ml/datasets/Iris.

[22] The Wine Dataset, https://archive.ics.uci.edu/ml/datasets/wine.