

Scientific Programming Approaches to Deep Learning for Source Code Transformation

Lead Guest Editor: Sikandar Ali

Guest Editors: Jiwei Huang, Zhongguo Yang, Muhammad Arif Shah,
Dayang Norhayati Abang Jawawi, and Najeeb Ullah





Scientific Programming Approaches to Deep Learning for Source Code Transformation

Scientific Programming

Scientific Programming Approaches to Deep Learning for Source Code Transformation

Lead Guest Editor: Sikandar Ali


Guest Editors: Jiwei Huang, Zhongguo Yang,
Muhammad Arif Shah, Dayang Norhayati Abang
Jawawi, and Najeeb Ullah



Copyright © 2021 Hindawi Limited. All rights reserved.

This is a special issue published in "Scientific Programming." All articles are open access articles distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Chief Editor



Emiliano Tramontana , Italy

Academic Editors

Marco Aldinucci , Italy
Daniela Briola, Italy
Debo Cheng , Australia
Ferruccio Damiani , Italy
Sergio Di Martino , Italy
Sheng Du , China
Basilio B. Fraguela , Spain
Jianping Gou , China
Jiwei Huang , China
Sadiq Hussain , India
Shujuan Jiang , China
Oscar Karnalim, Indonesia
José E. Labra, Spain
Maurizio Leotta , Italy
Zhihan Liu , China
Piotr Luszczek, USA
Tomàs Margalef , Spain
Cristian Mateos , Argentina
Zahid Mehmood , Pakistan
Roberto Natella , Italy
Diego Oliva, Mexico
Antonio J. Peña , Spain
Danilo Pianini , Italy
Jiangbo Qian , China
David Ruano-Ordás , Spain
Željko Stević , Bosnia and Herzegovina
Kangkang Sun , China
Zhiri Tang , Hong Kong
Autilia Vitiello , Italy
Pengwei Wang , China
Jan Weglarz, Poland
Hong Wenxing , China
Dongpo Xu , China
Tolga Zaman, Turkey

Contents


Frame Duplication Forgery Detection and Localization Algorithm Based on the Improved Levenshtein Distance

Honge Ren , Walid Atwa, Haosu Zhang, Shafiq Muhammad, and Mahmoud Emam 
Research Article (10 pages), Article ID 5595850, Volume 2021 (2021)

An Intelligent Analytics Approach to Minimize Complexity in Ambiguous Software Requirements

Fariha Ashfaq , Imran Sarwar Bajwa , Rafaqat Kazmi , Akmal Khan , and Muhammad Ilyas 
Research Article (20 pages), Article ID 6616564, Volume 2021 (2021)

Analyzing the Classification Techniques for Bulk of Cursive Languages Data: An Overview

Mu Hong , Shah Nazir , Zhang Shuo, and Wang Guan
Review Article (11 pages), Article ID 6624397, Volume 2021 (2021)


Software Birthmark Usability for Source Code Transformation Using Machine Learning Algorithms

Keqing Guan, Shah Nazir , Xianli Kong , and Sadaqat ur Rehman
Research Article (7 pages), Article ID 5547766, Volume 2021 (2021)



Applying Code Transform Model to Newly Generated Program for Improving Execution Performance

Bao Rong Chang , Hsiu-Fen Tsai , and Po-Wen Su 
Research Article (21 pages), Article ID 6691010, Volume 2021 (2021)



Key Performance Indicators for the Integration of the Service-Oriented Architecture and Scrum Process Model for IOT

Mengze Zheng , Islam Zada, Sara Shahzad, Javed Iqbal, Muhammad Shafiq, Muhammad Zeeshan, and Amjad Ali
Research Article (11 pages), Article ID 6613579, Volume 2021 (2021)




Analysis of Service-Oriented Architecture and Scrum Software Development Approach for IIoT

Yanqing Cui , Islam Zada, Sara Shahzad, Shah Nazir , Shafi Ullah Khan, Naveed Hussain, and Muhammad Asshad
Research Article (14 pages), Article ID 6611407, Volume 2021 (2021)

A Review on Multicriteria Decision Support System and Industrial Internet of Things for Source Code Transformation

Qinxia Hao , Shah Nazir , Xiaoxu Gao, Li Ma, and Muhammad Ilyas
Review Article (9 pages), Article ID 6661272, Volume 2021 (2021)

Software Piracy Awareness, Policy, and User Perspective in Educational Institutions

Zitian Liao , Shah Nazir , Anwar Hussain, Habib Ullah Khan , and Muhammad Shafiq
Research Article (14 pages), Article ID 6647819, Volume 2020 (2020)

Research Article

Frame Duplication Forgery Detection and Localization Algorithm Based on the Improved Levenshtein Distance

Hongre Ren ^{1,2} Walid Atwa,³ Haosu Zhang,² Shafiq Muhammad,⁴
and Mahmoud Emam ⁵

¹College of Information and Computer Engineering, Northeast Forestry University, Harbin 150040, China

²Heilongjiang Forestry Intelligent Equipment Engineering Research Center, Harbin 150040, China

³Department of Computer Science, Faculty of Computers and Information, Menoufia University, Shebin El-Koom 32511, Egypt

⁴Cyberspace Institute of Advance Technology, Guangzhou University, Guangzhou, China

⁵Department of Mathematics and Computer Science, Faculty of Science, Menoufia University, Shebin El-Koom 32511, Egypt

Correspondence should be addressed to Hongre Ren; nefu_rhe@163.com and Mahmoud Emam; ma7moud_emam@yahoo.com

Received 12 January 2021; Revised 3 March 2021; Accepted 20 March 2021; Published 1 April 2021

Academic Editor: Sikandar Ali

Copyright © 2021 Hongre Ren et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In this digital era of technology and software development tools, low-cost digital cameras and powerful video editing software (such as Adobe Premiere, Microsoft Movie Maker, and Magix Vegas) have become available for any common user. Through these softwares, editing the contents of digital videos became very easy. Frame duplication is a common video forgery attack which can be done by copying and pasting a sequence of frames within the same video in order to hide or replicate some events from the video. Many algorithms have been proposed in the literature to detect such forgeries from the video sequences through analyzing the spatial and temporal correlations. However, most of them are suffering from low efficiency and accuracy rates and high computational complexity. In this paper, we are proposing an efficient and robust frame duplication detection algorithm to detect duplicated frames from the video sequence based on the improved Levenshtein distance. Extensive experiments were performed on some selected video sequences captured by stationary and moving cameras. In the experimental results, the proposed algorithm showed efficacy compared with the state-of-the-art techniques.

1. Introduction

In our daily life, digital videos are playing a vital role in many fields of applications such as surveillance systems, medical fields, and criminal investigations. Because of the availability of low-cost digital video cameras and powerful video editing tools (such as Adobe Premiere, Microsoft Movie Maker, and Magix Vegas), it is now easy for common users to edit the video contents without leaving any visual traces of forgeries. So, we cannot trust in the authenticity of such videos anymore. Therefore, the authentication of such videos is becoming a very important research area these days. Digital video forensics is an emerging research area which aims at validating the authenticity of such videos [1]. The classification of digital video forensics is shown in Figure 1, where it can be divided

into 3 categories: identification of the source camera, discrimination of computer-generated videos, and video forgery detection (video tampering detection) [1].

Video forgery manipulations can be acted in the three domains: spatial domain (intraframe forgery), temporal domain (interframe forgery), and spatio-temporal domain. Intraframe forgeries may include region duplication (copy-move) and splicing inside the frame itself whereas interframe forgeries include frame duplication, frame insertion, frame shuffling, and frame deletion [1].

Digital video forgery detection algorithms aim to detect the traces of forgeries in the digital video sequence. As in digital images, digital video forgery detection techniques also can be classified into active and passive (blind) techniques. In the passive video forgery detection techniques, the authenticity of a forged video is verified without the existence

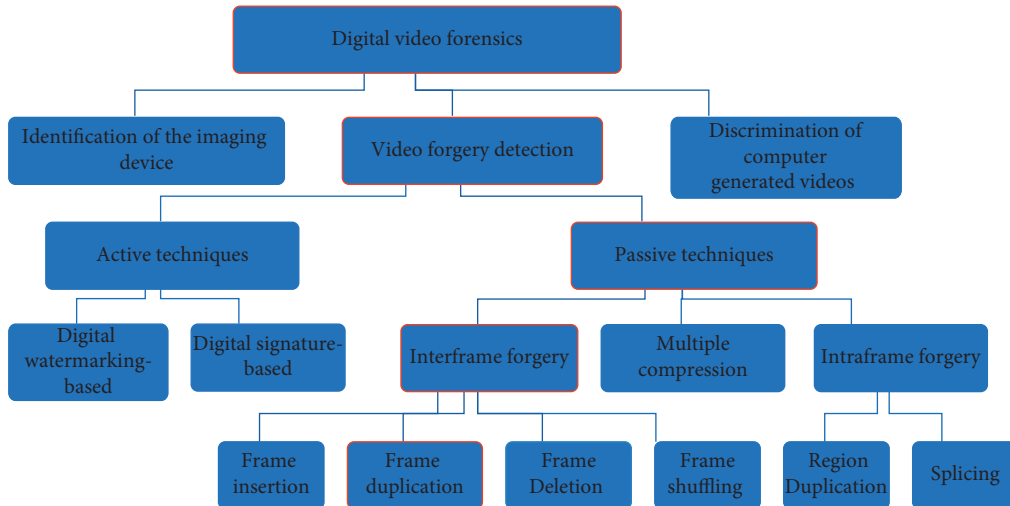


FIGURE 1: Digital video forensics and forgery detection techniques classification.

of the original video and only depends on extracting some features or footprints from the forged video which have been left by the editing operations [1]. These footprints may include the high spatio-temporal correlation among frame intensity values [2], noise and motion residues [3], artifacts in optical flow [4], motion-compensated edge artifact (MCEA) [5], and frames quality assessments [6] whereas active techniques require embedding information into the video such as digital watermarking [7], but this kind of techniques is not preferable by many researchers because it requires the existence of the original video along with the tampered one which is usually unavailable.

Frame duplication forgery is a common forgery types in digital videos, and it is an interframe forgery which occurs in the temporal domain. It can be performed by copying and pasting some frames in another location in the same video sequence in order to hide or replicate some events from the video. Figure 2 shows the process of frame duplication attack, where frames from 1 to 6 are copied and then pasted at another location instead of frames from 7 to 12 in order to remove the existence of a moving car crosses a parking area and passes behind a lamppost, without leaving any visual traces of forgeries. The original video example is taken from the LASIESTA dataset [8].

Cloning frames from the same video sequence raise the difficulty of frame duplication forgery detection, making it uneasy to detect color changes and illumination condition [9]. Although a variety of methods have been proposed, these methods still face the following challenges in frame duplication forgery:

- (1) High computational complexity
- (2) Low detection rate in the static scenes
- (3) Unable to locate the location of the duplicated frame pairs

In this paper, we proposed an efficient and robust frame duplication detection technique to detect duplicated frames from the video sequence based on the improved Levenshtein distance. At First, we divided the video sequence into small

overlapping subsequences and measure the similarity of them by using the improved Levenshtein distance (ILD). Next, the value of ILD is used to detect the duplication forgery frame, in which the higher the value, the lower the similarity between the frame pair. Hence finally, the duplicated frames are located. In the experimental results, the proposed algorithm showed efficacy compared with the state-of-the-art techniques.

The rest of this paper is organized as follows. In related work section, an overview of the related work and contributions in the field of frame duplication forgery detection are provided. Proposed Method section delineates the conceptual and implementation details of the proposed method. The experiments used during performance validation and the obtained results are discussed in Experimental Results section, and the paper is concluded in the last section.

2. Related Work

The frame duplication attacks can be detected from the tampered videos by using the existing digital image forgery detection techniques [10] as the video is a sequence of sequential images in one temporal (time t) and two spatial (x, y) dimensions. However, it may not seem a good idea due to the huge computational complexity obtained rather than the complex scenarios that the videos have such as static scenes [11]. Wang and Farid [12] proposed the first frame duplication forgery detection algorithm by using the spatial and temporal correlations between video frames. A coarse-to-fine comparison manner was used to compare the video subsequences. The high similarities in the temporal correlation coefficients lead to the spatial correlation coefficients comparison. However, their method was unable to detect the frame duplication forgeries in the static scenes and in case of postprocessing attacks such as adding noise on the duplicated frames. Using the previous framework, Yang et al. [6] proposed another two-stage similarity analysis-based method. In the first stage, they extracted the features from each video frame by using the singular value

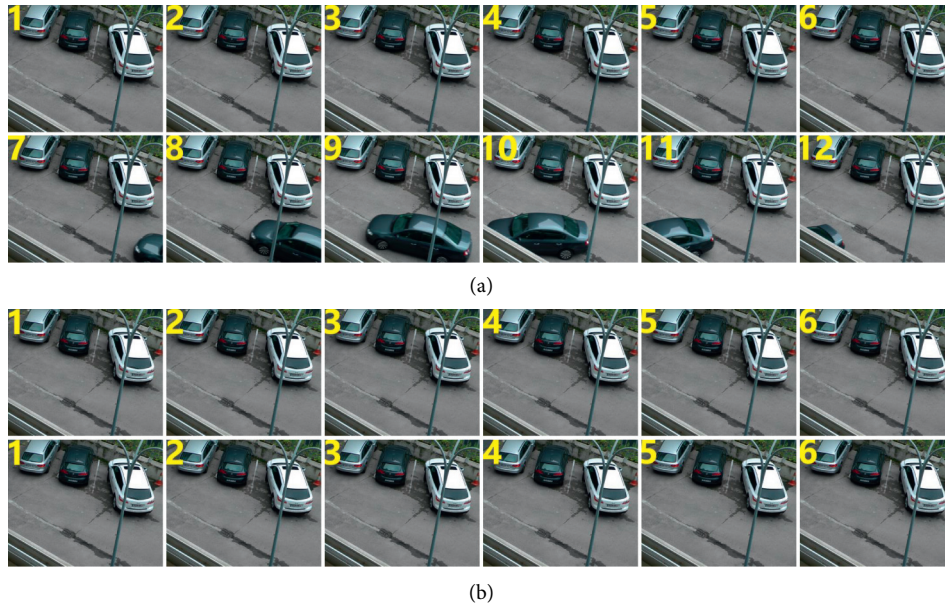


FIGURE 2: The process of frame duplication attack: an example. (a) The original video sequences. (b) The tampered video sequences.

decomposition (SVD). Then, the Euclidean distance similarity was calculated between the features of reference frame (first frame of the video) and each frame. In the second stage, a random block matching was used to indicate the candidate duplications. However, their method failed to detect the forgeries when frame duplications were done in different order and also when the duplicated frames were less than the window size [1]. Singh et al. [13] divided each video frame into four sub-blocks and then nine features were extracted from each frame. Then, they lexicographically sorted the extracted features to group the most similar frames. Root mean square error (RMSE) was then calculated between the features of adjacent sorted frames to identify the suspicious frames. Then, to detect the frame duplications, the correlation among suspicious frames was performed. Their method failed to detect the forged videos taken by a stationary camera and when duplication was done in different order [1]. Lin and Chang [14] presented an approach for frame duplication detection with four steps: candidate segment selection followed by spatial similarity measurement then frame duplication classification and finally postprocessing. However, many subsequence candidates were selected for the video that results in a significantly high computational time. Li and Huang [15] proposed another frame duplication forgery detection method based on the structural similarity (SSIM) [16]. The similarities between the video subsequences were calculated to find the duplicated frames. However, their method also failed to detect the frame duplication forgeries in the static scenes. D’Amiano et al. [17] proposed an algorithm for frame duplication forgery detection based on the dense-field method with invariant features. They used a suitable video-oriented version of patch-match to limit complexity. Jia et al. [9] proposed a novel approach to detect frame copy-move forgeries based on optical flow (OF), and stable parameters was designed.

The aforementioned methods used predefined fixed global thresholds during the candidates’ selection or

duplication detection stages. These thresholds may calibrate for a certain condition and may not work for other situations. Moreover, it makes these methods less generalized. Additionally, time complexity is one of the most challenging problems for frame duplication detection algorithms, which increases dramatically with increasing the number of frames within a given video sequence. Furthermore, there is inability to differentiate between the duplicated frame pairs and highly similar frame pairs (misdetected or false positive frame pairs) for the videos with long time static or still scenes.

3. Proposed Method

In interframe forgery (frame duplication), some frames from the video timeline are replaced by a copy from other frames from the same timeline (as shown in Figure 2). In this section, the proposed method for frame duplication forgery detection and localization is introduced in detail. The proposed method includes four stages as shown in Figure 3.

First, the video sequence is divided into small overlapping subsequences; second, similarity measurements based on Levenshtein distance [18] is calculated; third, frame duplication forgery is detected; and fourth, frame duplication forgery is located. To calculate the similarity between the video frames and identify the high similarities of the subsequences, the improved Levenshtein distances for all overlapping subsequences are calculated first. For the experiments, we tampered the video sequences with frame duplication forgery by randomly selecting the location in each video timeline.

3.1. Partition of Video Subsequence. In the experiments, the tampered video sequence \mathbf{V} is first divided into overlapping subsequences ($\mathbf{Seq}_i; i = 1, 2, 3, \dots, L$), which begin at time ζ .

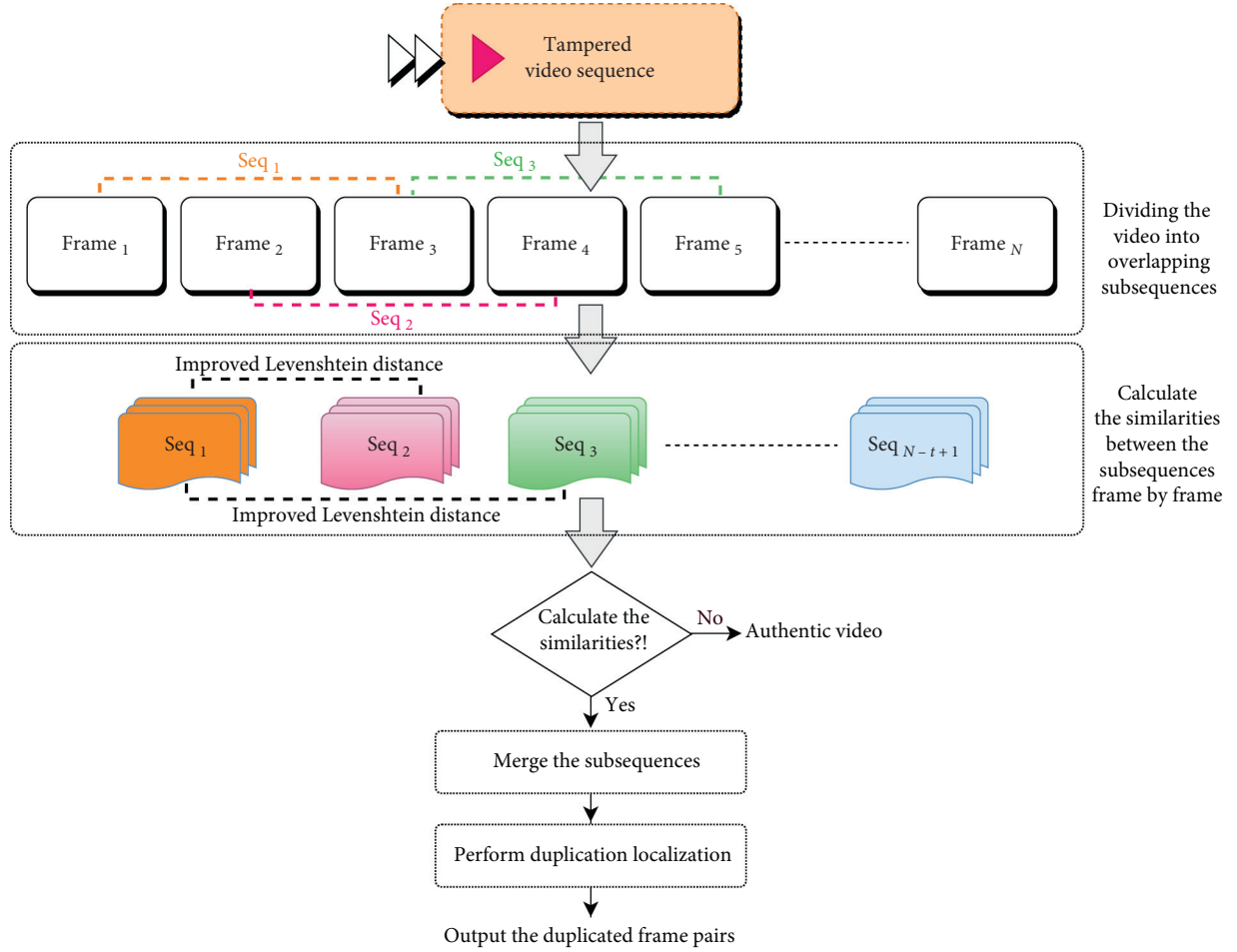


FIGURE 3: Proposed method steps.

L is the total number of all overlapping subsequences. We assumed that each subsequence length from the overlapping subsequences is (r) frames and the length of the test video is (N) frames. So, the total number of all overlapping subsequences (L) can be given by

$$L = N - r + 1; \quad 1 \leq r \leq N. \quad (1)$$

Next, we detect the potentially duplicated candidates by calculating the similarities among these subsequences. The similarity of each subsequence has to be calculated with the rest of the other subsequences. The improved Levenshtein distance is adopted and used as a measure of similarities in the proposed method, to measure the similarities between the corresponding frame pairs for each two candidates.

3.2. Similarity Measurements Based on the Improved Levenshtein Distance. The Levenshtein distance is a metric for measuring the similarities between two sets A and B as a simple function of their lengths ($|A|$ and $|B|$) [18]. The generalized Levenshtein distance (GLD) is the most common used measure to compare sets of different edit processes such as insertion, deletion, and substitution of sets elements [19]. The GLD can be obtained from the methods presented

in [18, 20]. It shows a distinct tool in some applications as error correction and pattern recognition [21, 22].

Assume that a pair of subsequences Seq_i and Seq_j from the tampered video V is denoted as $\text{Seq}_i = F_i^1 F_i^2 \dots F_i^r$ and $\text{Seq}_j = F_j^1 F_j^2 \dots F_j^r$, respectively, where F_i^m is the m th frame of Seq_i and F_j^n is the n th frame of Seq_j . The length of Seq_i is given by $|\text{Seq}_i|$. We set up the length of each subsequence to $r=5$ and the length of overlap is $r-1$.

$T_{F_i^m, F_j^n} = T_1 T_2 \dots T_l$ is used to show the edit transformation of F_i^m into F_j^n which is a sequence of elementary edit processes transforming F_i^m into F_j^n . Suppose an elementary edit process is (x, y) , if a weight function γ assigns to $x \rightarrow y$ a real number (non-negative) $\gamma(x \rightarrow y)$, the edit transformation weight $T_{F_i^m, F_j^n}$ can be computed by $\gamma(T_{F_i^m, F_j^n}) = \sum_{i=1}^r \gamma(T_i)$. Given F_i^m and F_j^n are the two frames from V , then the generalized Levenshtein distance (GLD) is calculated as in the following equation:

$$GLD(F_i^m, F_j^n) = \min \left\{ \gamma \left(T_{F_i^m, F_j^n} \right) \right\}. \quad (2)$$

If γ is a metric over the sequence of elementary edit processes, Marzal and Vidal in [23] defined the GLD as in the following equation:

$$GLD(\mathbf{F}_i^m, \mathbf{F}_j^n) = \min \left\{ W \left(P_{\mathbf{F}_i^m, \mathbf{F}_j^n} \right) \right\}, \quad (3)$$

where $P_{\mathbf{F}_i^m, \mathbf{F}_j^n}$ is an editing path between \mathbf{F}_i^m and \mathbf{F}_j^n and

$W(P_{\mathbf{F}_i^m, \mathbf{F}_j^n}) = \sum_{k=1}^{L(P_{\mathbf{F}_i^m, \mathbf{F}_j^n})} \gamma(F_{i_{k-1}+1 \dots i_k}^m \rightarrow F_{j_{k-1}+1 \dots j_k}^n)$ is the weight of $P_{\mathbf{F}_i^m, \mathbf{F}_j^n}$, which is a set of points or ordered pairs (i_k, j_k) , $0 \leq k \leq L(P_{\mathbf{F}_i^m, \mathbf{F}_j^n}) = l$ satisfying the following conditions:

- (1) $0 \leq i_k \leq |\mathbf{F}_i^m|$; $0 \leq j_k \leq |\mathbf{F}_j^n|$; $(i_0, j_0) = (0, 0)$; $(i_l, j_l) = (|\mathbf{F}_i^m|, |\mathbf{F}_j^n|)$
- (2) $\forall k \geq 1, 0 \leq i_k - i_{k-1} \leq 1$; $0 \leq j_k - j_{k-1} \leq 1$
- (3) $i_k - i_{k-1} + j_k - j_{k-1} \geq 1$

The improved Levenshtein distance similarity (ILD) is normalization for the GLD, and it can be easily computed through GLD. The improved Levenshtein distance between two frames \mathbf{F}_i^m and \mathbf{F}_j^n can be given as follows:

$$ILD(\mathbf{F}_i^m, \mathbf{F}_j^n) = \frac{2 \cdot GLD(\mathbf{F}_i^m, \mathbf{F}_j^n)}{\alpha \left(|F_i| + |F_j| \right) + GLD(\mathbf{F}_i^m, \mathbf{F}_j^n)}, \quad (4)$$

where $|F_i|$ and $|F_j|$ are the length of F_i and F_j , respectively. The final value of the improved Levenshtein distance calculated for two frames is included in the $[0, \infty)$ where 0 means that the two frames are identical (duplicated or replica) whereas any integer number between $1:\infty$ indicates the number of the different intensities in the corresponding frames.

To illustrate the advantages of ILD, we cut two consecutive frames representing a static scene in an authentic video, as shown in Figure 4. They have a high correlation coefficient that may cause misdetection. For example, the structural similarity (SSIM) between these two frames is calculated and it is equal to 0.9935, which means that if the threshold value in the SSIM-based algorithms [14, 15] is set to be smaller than or equal 0.9935, the detection performance of these frame duplication forgery detection algorithms will decrease dramatically due to the existence of falsely detected frame pairs as duplicated (misdetected frame pairs). Furthermore, they fail to detect the duplication in the static scenes whereas the improved Levenshtein distance between these two authentic frames is equal to 109, which means that there are 109 different pixel intensity values between these frames, and this indicates that these two frames are different and not a duplication from each other.

3.3. Merging Subsequences and Duplication Localization.

A helpful distance metric technique significantly improves the performance of localization, clustering, and classification processes [24]. Therefore, the distance metric techniques help algorithms to measure the similarities between the video contents. The tampered video sequence has been divided into small overlapping subsequences to detect frame duplication forgery. In order to form a set of candidate duplicated frames, several duplicated subsequences should be merged to form a complete duplicated sequence. Also, we need to identify which subsequence is original and which subsequence is duplicated (replica).

Due to the small overlapping subsequences, one or more subsequence could match with two or more duplicated subsequences. So, the subsequences with distances equal to 0 between their corresponding frame pairs are selected as a duplicated frame pairs to merge these subsequences in order to form a complete candidate subsequence of duplicated frames. In each subsequence, the similarities between each frame and all other frames of the other subsequences are calculated. Therefore, in this paper, we used the improved Levenshtein distance to calculate the similarities $\mathbf{D}[\mathbf{i}]$ among the corresponding candidate frames (f_i, f'_i) as follows:

$$D[\mathbf{i}] = \text{ILD}(f_i, f'_i), \quad (5)$$

where $i = 1, 2, \dots, r$.

Assume that (\mathbf{S}, \mathbf{T}) is a duplicated subsequence, \mathbf{S} and \mathbf{T} have the same number of frames (same length), and $(\mathbf{S}_i, \mathbf{T}_i)$ is a pair of corresponding matched frames. If all the ILD distances \mathbf{D}_i between $(\mathbf{S}_i, \mathbf{T}_i)$ are equal to 0, then \mathbf{S} is considered to be the source subsequence and \mathbf{T} is the duplicated one.

4. Experimental Results

4.1. The Dataset. In the experiments, we selected some test video sequences from the commonly used video test sequences from video trace library (VTL) dataset which is available at <http://trace.eas.asu.edu/yuv/index.html>. The selected videos are captured with stationary and moving camera modes. The resolution of each one is 352×288 pixels and has the frame rate of 30 fps. Table 1 shows the details of the test tampered videos.

4.2. Performance Evaluation and Analysis. The Precision and Recall rates are used as in equations (6) and (7) to evaluate the detection capability of the proposed method. We also calculate another measure F_1 score that combines both Precision and Recall as shown in equation (8).

$$\text{Precision} = \frac{T_P}{T_P + F_P} \times 100\%, \quad (6)$$

$$\text{Recall} = \frac{T_P}{T_P + F_N} \times 100\%, \quad (7)$$

$$F_1 \text{ score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}, \quad (8)$$

where T_P (true positive duplicated frame pairs) represents the number of correctly detected frame pairs as duplicated frames, F_P (false positive duplicated frame pairs) represents the number of falsely detected frame pairs as duplicated frames, and F_N (false negative duplicated frame pairs) represents the number of duplicated frame pairs which are classified as authentic.

To evaluate the performance of our proposed method, we compared our proposed method with Wang and Farid [12] and Li and Huang [15]. The Precision, Recall, and F_1 score rates are calculated for all of the forged videos in the

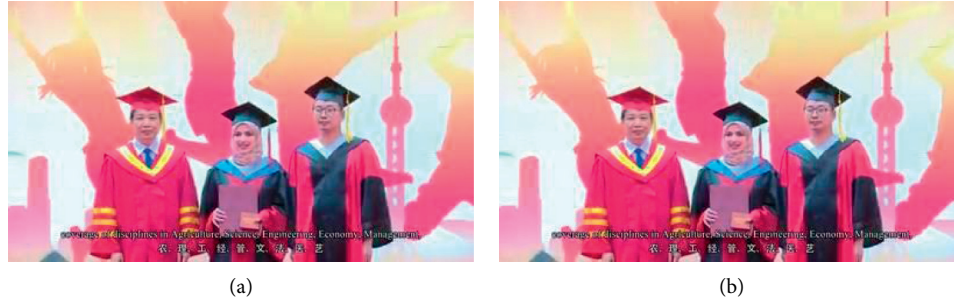


FIGURE 4: A comparison between SSIM and the improved Levenshtein distance for two consecutive frames of an authentic video. (a) Frame no. 2887. (b) Frame no. 2888. (SSIM=0.9935 and improved Levenshtein distance=109).

TABLE 1: Details of the selected test videos.

Video	Video frames	Video duration (sec)	Frame duplication location
Akiyo	300	10	1~20 are copied to 301~320
Bus	150	05	1~20 are copied to 151~170
Coastguard	300	10	1~20 are copied to 301~320
Container	300	10	1~20 are copied to 301~320
Flower	250	09	1~20 are copied to 251~270
Foreman	300	10	1~20 are copied to 301~320
Hall	300	10	1~20 are copied to 301~320
Mobile	300	10	1~20 are copied to 301~320
Silent	300	10	1~20 are copied to 301~320
Waterfall	260	09	1~20 are copied to 261~280

TABLE 2: Detection results of the proposed method for the tested video sequences.

Tampered video	Detection results		
	Proposed method	Wang and Farid [12]	Li and Huang [15]
Akiyo	Original: 1~20 Duplicates: 301~320	Authentic video	Original: 1~20 Duplicates: 301~320, with 192 misdetected frame pairs
Bus	Original: 1~20 Duplicates: 151~170	Original: 1~20 Duplicates: 151~170	Original: 1~20 Duplicates: 151~170
Coastguard	Original: 1~20 Duplicates: 301~320	Authentic video, with 6 misdetected frame pairs	Original: 1~20 Duplicates: 301~320
Container	Original: 1~20 Duplicates: 301~320	Authentic video	Original: 1~20 Duplicates: 301~320, with 39 misdetected frame pairs
Flower	Original: 1~20 Duplicates: 251~270	Duplicates: 251~270, with 61 misdetected frame pairs	Original: 1~20 Duplicates: 251~270
Foreman	Original: 1~20 Duplicates: 301~320	Duplicates: 301~320, with 17 misdetected frame pairs	Original: 1~20 Duplicates: 301~320
Hall	Original: 1~20 Duplicates: 301~320	Authentic video	Original: 1~20 Duplicates: 301~320
Mobile	Original: 1~20 Duplicates: 301~320	Duplicates: 301~320, with 407 misdetected frame pairs	Original: 1~20 Duplicates: 301~320

TABLE 2: Continued.

Tampered video	Proposed method	Detection results	
		Wang and Farid [12]	Li and Huang [15]
Silent	Original: 1~20	Original: 11~18	Original: 1~20
	Duplicates: 301~320	Duplicates: 311~318	Duplicates: 301~320, with 24 misdetected frame pairs
Waterfall	Original: 1~20	Original: 186~190	Original: 1~20
	Duplicates: 261~280	Duplicates: 193~197	Duplicates: 261~280

TABLE 3: Detection capability and location of duplication comparison.

Method	Precision (%)	Recall (%)	F_1 score (%)	Location of duplication
Wang and Farid [12]	28.34	44	34.47	No
Li and Huang [15]	78.88	100	88.19	Yes
Proposed method	99.50	100	99.75	Yes

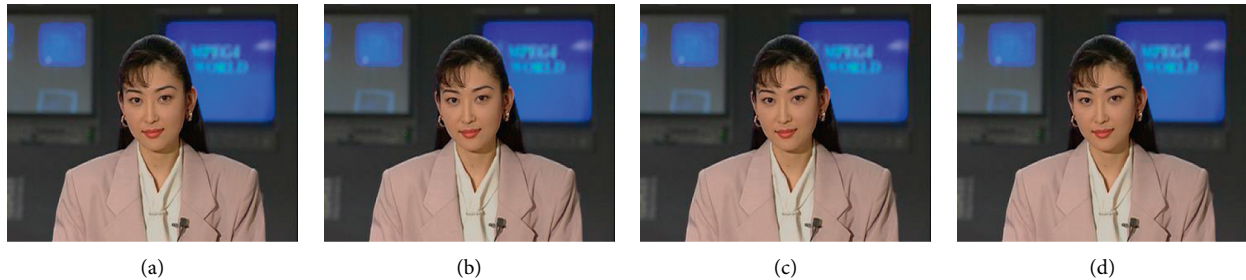


FIGURE 5: Snapshot for the first 4 frames in the test tampered video sequence (Akiyo).

dataset. The higher the Precision as well as the Recall rates and F_1 score are, the better performance will be.

Table 2 shows the detection results of the proposed method for the tested video sequences. It seems that the proposed method is not only able to achieve a high detection of frame duplication forgeries but also accurately locate the duplicated video clips in the video sequences. Table 3 indicates the comparison for the detection capabilities and location of duplication between the proposed method and the methods in [12, 15].

For the test tampered video Akiyo, the frames from 1:20 are duplicated in the location from 301:320. This video has a static (still) scene as shown in Figure 5, where the first four frames inside that video are visually the same (authentic frames-not duplicated). Figure 6 and Table 2 indicate that the proposed method is able to detect the frame duplication forgeries in the static scenes where the proposed method can correctly detect and locate the frame duplication forgeries (precision rate of 100%). However, the method in Wang and Farid [12] failed and identified this tampered video as an authentic video sequence whereas the method in Li and Huang [15] detected the frame duplication forgeries with low precision rate (9.43%) due to the existence of 192 misdetected frame pairs (false positive duplicated frame pairs). Therefore, the performance of our proposed method is much better than that of the other state-of-the-art methods in [12, 15], as shown in Tables 2 and 3 and Figure 6.

4.3. The Running Time. The comparison between the running time of the proposed method and the methods in Wang and Farid [12] and Li and Huang [15] is shown in Table 4. From that table, we can notice that the method proposed in Wang and Farid [12] has the lowest average time than others. The main reason is that the method proposed by Wang and Farid [12] was unable to locate the location of the duplicated frame pairs, which cost the other methods more time to localize the location of the duplicated frames. However, the method proposed in [12] has the worst detection accuracy than the other algorithms for frame duplication forgery detection (see Figure 6 and Table 2).

All the experiments are conducted on a workstation with Intel Core i7-8750H CPU and 32 GB RAM. We implemented the three methods on MATLAB R2018a.

Generally, the results presented in this paper reveal that the proposed algorithm offers a good performance in comparison with the state-of-the-art techniques. For the future directions, recently, deep learning approaches have been introduced in different fields of detection and identification problems [25–27]. It showed an efficacy and robustness against malicious attacks. Furthermore, copy-move forgery detection (CMFD) algorithms that have been presented for digital images can be used for video frame duplication forgery detection [28–30].

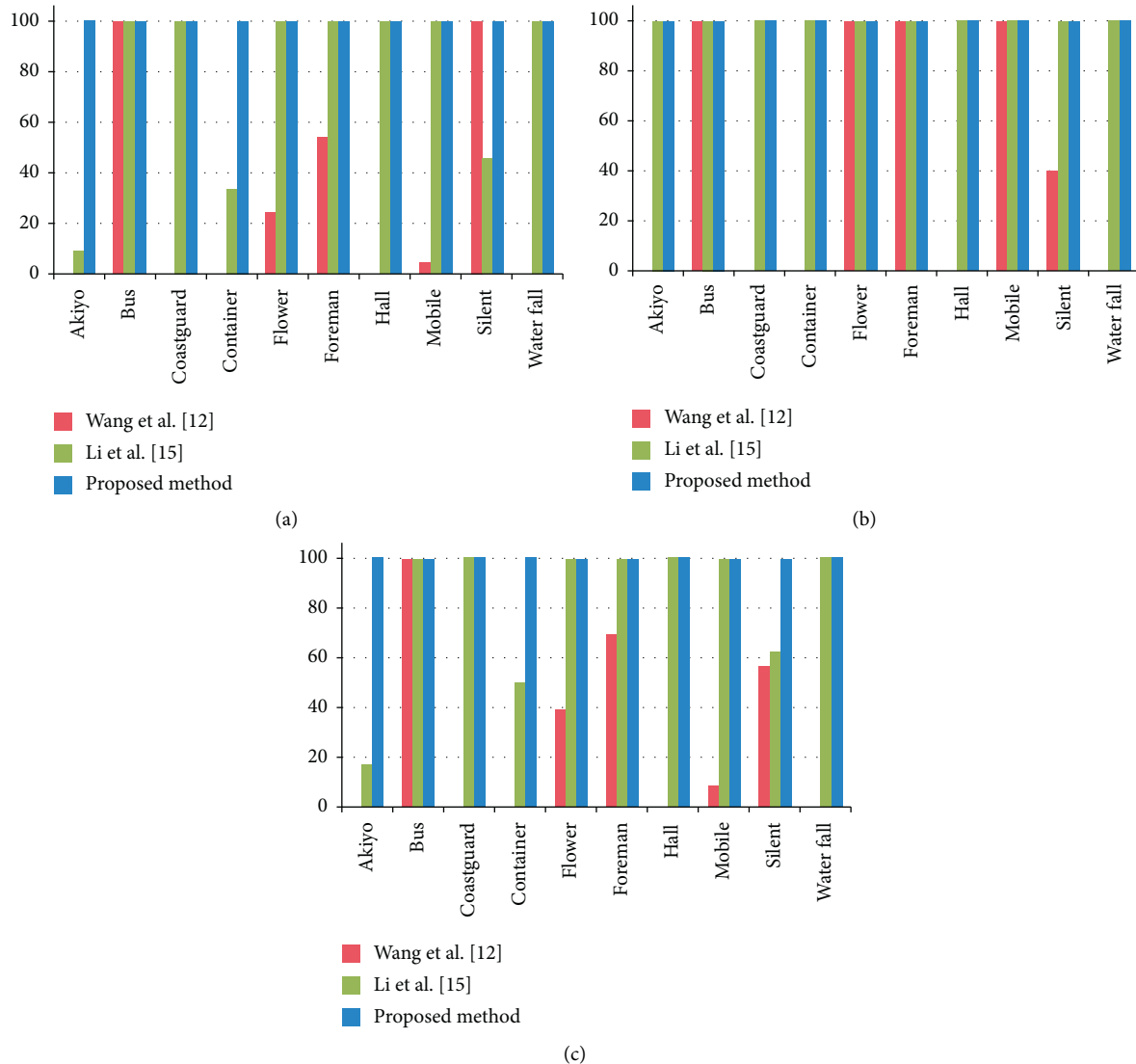


FIGURE 6: Comparison results with other algorithms. (a) Precision rates. (b) Recall rates. (c) F_1 score.

TABLE 4: The running time of each tampered video in the dataset.

Tampered video	Video frames (count)	Video duration (sec.)	Average running time (sec.)		
			Proposed method	Wang and Farid [12]	Li and Huang [15]
Akiyo	300	10	16.55	01.10	44.80
Bus	150	05	04.62	02.37	12.47
Coastguard	300	10	16.63	04.46	44.92
Container	300	10	16.40	01.06	45.53
Flower	250	09	11.68	09.12	32.12
Foreman	300	10	16.72	01.94	45.17
Hall	300	10	16.41	01.10	45.67
Mobile	300	10	16.46	14.09	45.66
Silent	300	10	16.17	01.22	45.80
Waterfall	260	09	12.57	0.93	35.01

5. Conclusion

This paper introduces a frame duplication forgery detection and localization approach based on the similarity analysis of

the improved Levenshtein distance. The tampered video sequence is first divided into overlapping subsequences. Next, each subsequence has to calculate the similarities with the rest of the other subsequences. The improved

Levenshtein distance is adopted and used as a measure of similarities in this paper. The similarities between all the subsequences are measured to find out the potentially duplicated frame pairs. These duplicated frame pairs are combined together into a complete duplicated sequence, and hence the location of the frame duplication forgeries is located. Extensive experiments are conducted on some tampered videos downloaded from VTL dataset. The results show that the precision of the proposed method can achieve 99.5% which is higher than the state-of-the-art methods. Furthermore, the proposed method is able to locate the exact location of the replica in addition to the detection capabilities of frame duplication forgeries from the static scenes.

Data Availability

No private data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Fundamental Research Funds for the Central Universities under grant nos. 2572018BH09 and 2572017PZ10 and Postdoctoral Research Program of Northeast Forestry University under grant no. 203822.

References

- [1] K. Sitara and B. M. Mehtre, "Digital video tampering detection: an overview of passive techniques," *Digital Investigation*, vol. 18, pp. 8–22, 2016.
- [2] G.-S. Lin, J.-F. Chang, and C.-H. Chuang, "Detecting frame duplication based on spatial and temporal analyses," in *Proceedings of the 2011 6th International Conference on Computer Science & Education (ICCSE)*, pp. 1396–1399, Singapore, August 2011.
- [3] R. C. Pandey, S. K. Singh, and K. K. Shukla, "Passive forensics in image and video using noise features: a review," *Digital Investigation*, vol. 19, pp. 1–28, 2016.
- [4] W. Wang, X. Jiang, S. Wang, M. Wan, and T. Sun, "Identifying video forgery process using optical flow," in *Proceedings of the 12th International Workshop on Digital-Forensics and Watermarking (IWDW)*, pp. 244–257, Auckland, New Zealand, October 2013.
- [5] Q. Dong, G. Yang, and N. Zhu, "A MCEA based passive forensics scheme for detecting frame-based video tampering," *Digital Investigation*, vol. 9, no. 2, pp. 151–159, 2012.
- [6] J. Yang, T. Huang, and L. Su, "Using similarity analysis to detect frame duplication forgery in videos," *Multimedia Tools and Applications*, vol. 75, no. 4, pp. 1793–1811, 2016.
- [7] Y. Shi, M. Qi, Y. Yi, M. Zhang, and J. Kong, "Object based dual watermarking for video authentication," *Optik*, vol. 124, no. 19, pp. 3827–3834, 2013.
- [8] C. Cuevas, E. M. Yáñez, and N. García, "Labeled dataset for integral evaluation of moving object detection algorithms: Lasiesta," *Computer Vision and Image Understanding*, vol. 152, pp. 103–117, 2016.
- [9] S. Jia, Z. Xu, H. Wang, C. Feng, and T. Wang, "Coarse-to-fine copy-move forgery detection for video forensics," *IEEE Access*, vol. 6, pp. 25323–25335, 2018.
- [10] O. M. Al-Qershi and B. E. Khoo, "Passive detection of copy-move forgery in digital images: state-of-the-art," *Forensic Science International*, vol. 231, no. 1–3, pp. 284–295, 2013.
- [11] A. Bovik, "Handbook of image and video processing," *Sensor Review*, vol. 62, no. 4, pp. 4632–4636, 2005.
- [12] W. Wang and H. Farid, "Exposing digital forgeries in video by detecting duplication," in *Proceedings of the 9th Workshop on Multimedia & Security*, pp. 35–42, Dallas, TX, USA, September 2007.
- [13] V. K. Singh, P. Pant, and R. C. Tripathi, "Detection of frame duplication type of forgery in digital video using sub-block based features," in *Proceedings of the International Conference on Digital Forensics and Cyber Crime*, pp. 29–38, Seoul, Korea, October 2015.
- [14] G.-S. Lin and J.-F. Chang, "Detection of frame duplication forgery in videos based on spatial and temporal analysis," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 26, no. 7, p. 1250017, 2012.
- [15] F. Li and T. Huang, "Video copy-move forgery detection and localization based on structural similarity," in *Proceedings of the 3rd International Conference on Multimedia Technology (ICMT 2013)*, pp. 63–76, Springer, Berlin, Germany, 2014.
- [16] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.
- [17] L. D'Amiano, D. Cozzolino, G. Poggi, and L. Verdoliva, "A patch match-based dense-field algorithm for video copy-move detection and localization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 3, pp. 669–682, 2018.
- [18] K. N. S. Behara, A. Bhaskar, and E. Chung, "A novel approach for the structural comparison of origin-destination matrices: Levenshtein distance," *Transportation Research Part C: Emerging Technologies*, vol. 111, pp. 513–530, 2020.
- [19] J. Beernaerts, E. Debever, M. Lenoir, B. De Baets, and N. Van de Weghe, "A method based on the Levenshtein distance metric for the comparison of multiple movement patterns described by matrix sequences of different length," *Expert Systems with Applications*, vol. 115, pp. 373–385, 2019.
- [20] W. J. Masek and M. S. Paterson, "A faster algorithm computing string edit distances," *Journal of Computer and System Sciences*, vol. 20, no. 1, pp. 18–31, 1980.
- [21] J. L. Peterson, "Computer programs for detecting and correcting spelling errors," *Communications of the ACM*, vol. 23, no. 12, pp. 676–687, 1980.
- [22] G. Navarro, "A guided tour to approximate string matching," *ACM Computing Surveys*, vol. 33, no. 1, pp. 31–88, 2001.
- [23] A. Marzal and E. Vidal, "Computation of normalized edit distance and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 9, pp. 926–932, 1993.
- [24] D. A. Adjeroh, M.-C. Lee, and I. King, "A distance measure for video sequences," *Computer Vision and Image Understanding*, vol. 75, no. 1–2, pp. 25–45, 1999.
- [25] M. Shafiq, Z. Tian, A. K. Bashir, X. Du, and M. Guizani, "Corrauc: a malicious Bot-IoT traffic detection method in IoT network using machine learning techniques," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3242–3254, 2020.
- [26] M. Shafiq, Z. Tian, Y. Sun, X. Du, and M. Guizani, "Selection of effective machine learning algorithm and Bot-IoT attacks

- traffic identification for internet of things in smart city,” *Future Generation Computer Systems*, vol. 107, pp. 433–442, 2020.
- [27] M. Shafiq, Z. Tian, A. K. Bashir, X. Du, and M. Guizani, “IoT malicious traffic identification using wrapper-based feature selection mechanisms,” *Computers & Security*, vol. 94, p. 101863, 2020.
- [28] M. Emam, Q. Han, and H. Zhang, “Detection of copy-scale-move forgery in digital images using SFOP and MROGH,” in *Proceedings of the International Conference of Pioneering Computer Scientists, Engineers and Educators*, pp. 326–334p, Harbin, China, August 2016.
- [29] M. Emam, Q. Han, L. Yu, Y. Zhang, and X. Niu, “A passive technique for detecting copy-move forgery with rotation based on polar complex exponential transform,” in *Proceedings of the Seventh International Conference on Digital Image Processing (ICDIP 2015)*, Los Angeles, CA, USA, April 2015.
- [30] M. Emam, Q. Han, L. Yu, and H. Zhang, “A keypoint-based region duplication forgery detection algorithm,” *IEICE Transactions on Information and Systems*, vol. E99.D, no. 9, pp. 2413–2416, 2016.

Research Article

An Intelligent Analytics Approach to Minimize Complexity in Ambiguous Software Requirements

Fariha Ashfaq ¹, **Imran Sarwar Bajwa** ¹, **Rafaqut Kazmi** ¹, **Akmal Khan** ¹,
and Muhammad Ilyas ²

¹Department of Computer Science, The Islamia University of Bahawalpur, Bahawalpur 63100, Pakistan

²Department of Computer Science, University of Malakand, Chakdara, Pakistan

Correspondence should be addressed to Imran Sarwar Bajwa; imran.sarwar@iub.edu.pk

Received 16 November 2020; Revised 13 January 2021; Accepted 8 March 2021; Published 23 March 2021

Academic Editor: Fabrizio Riguzzi

Copyright © 2021 Fariha Ashfaq et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

An inconsistent and ambiguous Software Requirement Specification (SRS) document results in an erroneous/failed software project. Hence, it is a serious challenge to handle and process complex and ambiguous requirements. Most of the literature work focuses on detection and resolution of ambiguity in software requirements. Also, there is no standardized way to write unambiguous and consistent requirements. The goal of this research was to generate an ambiguity-less SRS document. This paper presents a new approach to write ambiguity-less requirements. Furthermore, we design a framework for Natural Language (NL) to Controlled Natural Language (CNL) (such as Semantic Business Vocabulary and Rules (SBVR)) transition and develop a prototype. The prototype also generates Resource Description Framework (RDF) representation. The SBVR has a shared meaning concept that minimizes ambiguity, and RDF representation is supported by query language such as SPARQL Protocol and RDF Query Language (SPARQL). The proposed approach can help software engineers to translate NL requirements into a format that is understandable by all stakeholders and also is machine processable. The results of our prototype are encouraging, exhibiting the efficient performance of our developed prototype in terms of usability and correctness.

1. Introduction

Requirements engineering (RE) has been dealing with similar issues since the beginning in software engineering. RE is a common phase in business process development as well as software development for user requirement specification. Typically, the four stages of RE in the software development process include requirement elicitation, analysis, requirement validation, and specification. Here, requirement elicitation and requirement analysis phases are significant to ensure complete, consistent, and unambiguous requirement specifications. The end product of the RE phase is the SRS document. The SRS document becomes the base of the later stages [1] of the software development process. The initial phases of the software development process have more impact on the software quality as compared to later phases [2]. The SRS document maintains user end story in descriptive form using NL [1], because NL is the most

convenient way to communicate while gathering requirements in business process development as well as in software development. Almost 79% of requirement specification documents are found in “Real Natural Language” [3]. NL is expressive, universal, flexible, widespread, and above all understandable for all stakeholders [4]. Along with the key features, NL is inherently ambiguous [5] and has extensively been recognized as a challenge [6].

In The Ambiguity Handbook, Berry et al. [7] define ambiguity as, “A requirement is ambiguous if it admits multiple interpretations despite the reader’s knowledge of the requirement engineering context.” There are 39 types of ambiguity, vagueness, or generality [7]. NL such as English is ambiguous owing to its informal sentence construction. There exists an average of 23 meanings for the 500 most used English words [8]. Ambiguity is more intractable than other requirement defects and, thus, results in more frequent misunderstandings [9].

If the SRS document contains ambiguous, incomplete, or contradictory requirements, it results in erroneous software. According to S. McConnell, more than 50% of corporate software projects do not fulfill the expectations [10]. CHAOS report of Standish Group 2015 states that 52% of software exceed budget as well as time or does not meet the basic requirements, while 19% of projects are a complete failure [11]. All such failures are due to misunderstandings while interpreting stakeholder's ambiguous requirements. The SRS document is considered unambiguous only if there is only one possible meaning for each software requirement [12]. The SRS document contains requirements understood and agreed upon by all stakeholders [13].

Inconsistency and ambiguity in the SRS document propagate to the next phases of software development [14], such as software modeling; affect the accuracy of the end software product; and results in more manual effort and high-cost budgets. Hence, the functional requirements gathered and quoted in the SRS document must be complete, consistent, and clear, i.e., unambiguous.

A possible solution to handle ambiguity can be the exercise of a mathematical formal logic illustration instead of NL to represent user requirements. However, the use of formal logic is not only a complex task. A wrongly written formal logic will be difficult to handle, and it will create a serious problem in later stages of software development. Furthermore, stakeholders are usually not able to understand mathematical logic. Hence, this solution does not seem feasible.

Another possible way to deal with the above-discussed ambiguity problem is the use of CNL [15]. It can work as a bridge between NL and formal representation. Since Requirement Analysis is based on communication and the analyst's experience, it can be modeled up to a certain limit. This limit gives birth to controlled language. If the document is written in CNL, it will be feasible for a development team to use a simpler and less costly linguistic tool. Several CNL could be found in literature such as ACE [16], PENG [17], CPL [18], Formalized-English [19], SBVR, etc. We intend to apply an SBVR-based CNL to write stakeholder's requirements and generate an SRS document. Such software requirements will be syntactically unambiguous as well as semantically consistent. In this paper, a novel approach is presented to soften the complex requirements by representing the requirements in standardized forms such as SBVR and RDF. By "soften the complex requirement," we mean to rewrite a complex requirement in such a standardized form so that the ambiguity caused by the complexity of the sentence can be avoided.

2. Literature Review

Two classes' approach is found in RE to handle ambiguity [13]. The first-class reactive approaches include ambiguity detection and resolution. Such class deals tend to base on cases where ambiguity is already present in the SRS document's text. The second class is proactive, i.e., ambiguity prevention/avoidance deals with ambiguity in the first place, i.e., the requirement elicitation phase. It detects ambiguity in

requirements and demands an explanation from the user. The five major approaches in which research work is being done to deal with both classes of ambiguity include checklist-based inspection approach; style guides; knowledge-based approach; heuristics-based approach; and controlled language [20]. Bano [6] and Sandhu and Sikka [18] have conducted a deep survey on ambiguity handling approaches.

In a literature review [6], the author argued that there is more focus on ambiguity detection, whereas ambiguity avoidance and resolution have been largely neglected in empirical work. Also, more work is done on syntactic and semantic ambiguities than other types. In literature, different methods and techniques are proposed to tackle ambiguity in the SRS document [6, 20].

Friedrich et al. [21] present an automatic approach to produce Business Process Model and Notation models from the NL text. The purpose is to minimize ambiguities and time. The researchers combined existing tools from NLP and improved them using an appropriate anaphora resolution mechanism. The generated SBVR specifications are validated and verified using the SBVR2UML approach proposed by [22]. This tool translates SBVR specifications documented in the SRS document to Unified Modeling Language (UML) class diagram. The technique extracts Object-Oriented information from SBVR-SRS and maps it to a class model.

RESI [23] is an ontology-based "common sense engine" supporting analysts by providing a dialog-system to make suggestions and inquires for an ambiguous, faulty, or inaccurate specification. RESI works in four steps, and utilizes various ontologies such as ResearchCyc, WordNet, ConceptNet, and YAGO to discover problems and to provide common-sense solutions. Mich and Garigliano [24] have proposed a scheme to define indices of structural and semantic ambiguity. Researchers have investigated the application of defined indices for the system called LOLITA. It uses the knowledge base (a kind of conceptual graph) and output of the parsing phase to calculate the feasibility of these indices. LOLITA is designed to prevent ambiguity. LOLITA detects structural and semantic ambiguity.

A tool SREE [25] is designed to detect instances of potential ambiguity in NL requirement specifications and reports them back to the user to take appropriate action. SREE uses a lexical analyzer, searching for instances of only precise words in the SREE's database. The tool uses guiding rules that help to write less ambiguous NL SRS documents. The provided guiding rules can also serve as an inspection checklist to find ambiguities in requirement specifications. SREE only detects coordination ambiguity and semantic ambiguity. Yang et al. [26] constructed a machine-learning-based antecedent classifier, to detect anaphoric ambiguities. The antecedent classifier trains itself on a set of heuristics and human judgments. The output is then used to spot noxious ambiguity. The used classifier alerts the requirements analyst about the risk of misinterpretation. A Comparative Analysis is mentioned in Table 1.

Instead of generating user understandable format of the SRS document, research focused on NL to model/prototype generation. All of the abovementioned tools motivate us to

TABLE 1: Comparative analysis of work done on ambiguity.

Sr #	Source	Data type	Identified ambiguities	Proposed approach	Technologies/models/methods/algorithms used	Ambiguity avoidance	Ambiguity detection	Ambiguity resolution
1	Kamsties et al. [27]	NL: English	Lexical, Polysemy, systematic Polysemy, Referential Discourse, Domain	Checklist-based inspection approach UML-based heuristics-based approach	UML Foundation package, SCR metamodel	✗	✓	✗
2	Mich and Garigliano [24]	NL: English	Semantic Syntactic	Knowledge base, NLP, indices	LOLITA	✗	✓	✗
3	Ashfa and Imran Sarwar Bajwa [11]	NL: English office time management system	Syntactic Semantic	SBVR, controlled NL	Stanford POS tagger v3.0rule-based bottom-up parser	✓	✗	✗
4	Ferrari et al. [28]	NL: EnglishOutbreak Management Functional Requirements, issued in 2005 by the Public Health Information Network (PHIN) of the Centers for Disease Control and Prevention (CDC)	Pragmatic	Knowledge-based approach	Least-cost path search	✗	✓	✗
5	Friedrich et al. [21]	NL:47 text-model pairs from industry and textbooks	Referential anaphora resolution	NLP-based	Stanford Parser, FrameNet, WordNet, World Model, the anaphora resolution algorithm	✗	✓	✗
6	Kaiya and Saeki [29]	NL: Japanese, software music player	Semantic	Ontology-Based	Spreadsheet, a diagram editor with macro processing	✗	✗	✗
7	Gleich et al. [30]	NL:data set consisting of approximately50 German and 50 English sentences	Lexical Syntactic Semantic Pragmatic	Automated ambiguity detection, NLP-based	Unix-based grep-like technique	✗	✓	✗
8	Al-Harbi et al. [31]	NL: English200 NL questions for the university domain from various universities' websites	Lexical. semantic (nouns-based ambiguity)	Context knowledge and concepts ontology, shallow NL processing based	Rule-based chunker, rule-based shallow semantic analyzer, semantic role labeling method, WordNet Domains	✗	✗	✓
9	Verma and Beg [32]	NL: English	Syntactic Incompleteness	NLP-based	Shift-reduce style parser, maximum entropy parser, Penn Tree Bank NL Processing, Word Sense Disambiguation, Text Mining	✗	✗	✓
10	Gill et al. [33]	NL: English	Lexical Syntactic Semantic	NLP-based	Language Perceptions, Human Judgment, Contextual Knowledge	✓	✗	✗

TABLE 1: Continued.

Sr #	Source	Data type	Identified ambiguities	Proposed approach	Technologies/models/methods/algorithms used	Ambiguity avoidance	Ambiguity detection	Ambiguity resolution
11	Massey et al. [34]	NL: English (23 paragraphs from HITECH Act, 45 CFR Subtitle A, § 170.302)	Lexical, Syntactic Semantic, Vagueness Incompleteness, Referential	Manual guide-based	Case Study Ambiguity Taxonomy	✗	✓	✗
12	Sandhu and Sikka [18]	NL: English	Lexical Syntactic Semantic Pragmatic Pragmatic Semantic	Knowledge-based approach	Rule-based framework	✗	✓	✗
13	Bhatia et al. [5]	NL: English	Vagueness and Generality, Language error	Knowledge-based ontology-based	Ontology-based framework	✗	✓	✗
14	Sabriye and Zainon [1]	NL: English Open-source software requirements specification documents	Syntactic Syntax	NLP-based	Stanford POS tagger, rule-based ambiguity detector	✗	✓	✗
15	Ali et al. [35]	NL: English Services Proz: workforce software	Syntactic Syntax	NLP-based ontology-based	IEEE 830 Template, RUP template, Pragmatic Quality Model (PQM), and Perspective-based Reading (PBR)	✗	✓	✓
16	Popescu et al. [36]	NL SRS	Semantic	Controlled NLP-based	Dowser parser	✗	✓	✗

design an approach that can assist analysts in creating ambiguity-less SRS documents from English. The above-mentioned tool tends to deal with a certain type of ambiguity, thus others remain a concern. Along with syntactic and semantic ambiguities, there is a need to work on other types of ambiguities as well.

To minimize the effect of ambiguity and informality of NL in the SRS document, the best approach is to use some sort of combined approach. CNL (a subset of NL) such as Attempto [37] and SBVR [38] minimize formality as well as help to generate ambiguity-less SRS document. For business process modeling, SBVR has emerged as a vital tool for capturing software requirements. It is an efficient way to get machine processable and unambiguous rules written in a formal pattern from NL.

Most of the work is based on the reactive approach; previous solutions tend to base on cases where ambiguity is already present in the text. Most of the works deal with ambiguity detection. There is a huge gap in research to find a way to avoid ambiguity in the first place. Moreover, the proposed solution deals with a certain type of ambiguities. There is a lack of a standard approach that successfully deals with all types of ambiguities. Bajwa et al. [39] have proposed an automated approach, NL2SBVR. NL2SBVR uses the UML class model as a business domain to map NL to SBVR vocabulary. It translates NL specifications to formal SBVR rules. The SBVR provides a set of logical formulations that can help in the semantic analysis of English language text. NL2SBVR will give efficient results in less time and with

lesser errors. The proposed approach achieved an average accuracy of 87.33%. The approach lacks a method to specify the meaning for entities in the dataset.

Ramzan et al. [40] developed a tool for NL to SBVR model transformation using the Sitra transformation engine. To accomplish the transformation, the concept of English metamodel is introduced. The transformation is based on SBVR 1.0 document. The proposed approach achieved an average recall of 83.22%, average precision of 87.13%, and average *F*-value of 85.14. Danenas et al. [41] propose the M2M conversion. Authors extract data from UML case diagrams using Text rumblings. Preprocessing is performed on the extracted result of Text rumblings. In the end, the SBVR model is generated. A common understanding of things is the prerequisite for nonautomated parts and, thus, can cause ambiguity.

Njonko and El Abed [42] transform NL business rules into SBVR Rules using the NLP framework. SBVR rules are then transformed into Executable Models. The evaluation of the proposed approach is missing. Hence, the success factor of the proposed approach is unknown. Siqueira et al. [43] combine the Behavior-Driven Development approach and SBVR to refine requirements through iterations. Chittimalli and Anand [44] check the inconsistencies among SBVR rules using Satisfiability modulo theories. Arnold and Rahm [45] propose an approach to identify the semantic similarity between concepts from Wikipedia articles.

We propose a proactive approach to handle all the parameters that actively play a role in raising ambiguities in

NL software requirements. Such a proactive approach focuses on representing software requirements, represented in NL, in a CNL such as SBVR that is based on formal logic. It provides a shared meanings concept. The output of our proposed approach will not only be ambiguity-less and understandable by all stakeholders but also machine processable at the same time. Table 2 presents a comparative analysis between some existing tools for ambiguity prevention and the proposed approach.

2.1. Semantic Business Vocabulary and Rules (SBVR). The SBVR was first released by Object Management Group (OMG) in 2008. The SBVR can define clear, syntactically, and semantically unambiguous, meaning-centric business vocabulary and rules not only in business but also in the software industry. The SBVR has mathematical foundations (first-order logic), which make it easy to be processed by a machine. The SBVR's English-like syntax makes it easy to understand for all stakeholders.

The SBVR has the same expressive power as standard ontological languages [24]. The produced SBVR XML schema makes the interchange of information easy among organizations and software tools. In the end, the use of the SBVR will reduce the overall cost of the software development process. The SBVR 1.4 [38] is the latest standard by OMG. The SBVR vocabulary (Concepts, Fact Types) and rules constitutes a standardized SBVR representation [38].

2.2. How the SBVR Can Help to Eliminate Ambiguity? The SBVR vocabulary is not only a set of definitions of terms (concepts) and other representations but is an organized set of interconnected concepts. For a particular business domain, the SBVR vocabulary defines each term with one exact meaning in a given context, thus eliminating the chance of ambiguity. Such definitions are formal enough to be used by other software tools and informal enough to be understood and used by all stakeholders.

2.2.1. Removing Lexical Ambiguity. In the SBVR, an expression represents a concept. Each expression/wording is uniquely associated with one meaning (concept) in a given context. Thus, one concept–one meaning association eliminates the chance of lexical ambiguity. The definition of a concept incorporates delimiting (a concept must hold) and implied (a concept must not hold) characteristics, eliminating the chances of homonymy lexical ambiguity. Along with the definition, each concept has a definite description along with examples and notes, hence removing polysemy ambiguity. By following the SBVR rules to create a vocabulary, the result will be well-defined concepts that cannot be taken in multiple senses within the community, thereby eliminating all types of lexical ambiguities.

2.2.2. Removing Syntactic Ambiguity. The SBVR includes “Verb Concept Wording” to formally specify syntactic representations of concepts and rules of any domain in NL. Such a feature makes the SBVR well-suited for

describing syntactically ambiguity-less software requirements. The SBVR identifies concepts along with the grammatical roles they play in a certain situation. Such a feature removes the analytical ambiguity from a given requirement. For each verb concept, the SBVR identifies related categorization, classification, characterization, and situational roles. Such identification will minimize attachment ambiguity. Furthermore, the SBVR uses logical formulation such as logical operations and, thus, eliminates the coordination ambiguity.

2.2.3. Removing Semantic Ambiguity. The SBVR provides semantic formulations to make English statements controlled and semantically ambiguity-less. Such a semantic formulation includes atomic formulation, instantiate formulation, modal formulation, logical formulation, quantification, objectification, and nominalizations. The use of logical formulation such as quantification eliminates the scope of ambiguity. The SBVR incorporates “Reference Scheme.” Such schemes serve as a link for noun phrases and prepositions to their corresponding concepts and, thus, eliminate Referential Ambiguity.

2.2.4. Removing Pragmatic Ambiguity. The SBVR avoids pragmatic ambiguity by identifying associative fact types. The binary fact type is a typical example of the Associative Fact Type. In the example “The car conveys the parts,” there is a binary association between the car and parts concepts. This association is one-to-many as the “parts” concept is plural. In the conceptual modeling of the SBVR, Associative Fact Types are mapped to associations.

3. Proposed Methodology

The main objective of this research is to prevent the SRS document from inducting ambiguity in the first place. To achieve this, the proposed methodology comprises four main phases, i.e., preliminary investigation and analysis, proposed framework design and implementation, data collection and experimental evaluation, and research findings and conclusion.

3.1. Preliminary Investigation and Analysis. In the first phase of the research, existing literature associated with the study has been studied. The purpose is to acquire knowledge related to RE techniques—the use of NL in SRS documents and the ambiguity caused by NL. Possible solutions in the literature to handle ambiguity in NL have been studied.

3.2. Proposed Approach Design and Implementation. Based on preliminary investigation and analysis, an ambiguity prevention approach is proposed to avoid ambiguity. A prototype is developed to assess the anticipated approach.

3.3. Data Collection and Experimental Evaluation. In the third phase, the data for the evaluation of the approach will

TABLE 2: A comparative analysis of existing tool with the presented approach.

Feature Support	Korner and Brumm [23]	Mich and Garigliano [24]	Ashfa, and Imran Sarwar Bajwa [11]	Al-Harbi et al. [31]	Verma and Beg [32]	Our proposed approach
Approach used	Knowledge-based to ontology	Knowledge base	Controlled language	Ontology	NLP	Controlled language
Technologies/ Models/Methods/ algorithms/ Approach used	RESI Stanford parser, ConceptNet, WordNet	LOLITA Indices	SR-Elicitor SBVR, Stanford POS tagger rule-based bottom-up parser	Shift-reduce style parser, maximum entropy parser, Penn Tree Bank	NL Processing, Word Sense Disambiguation	SBVR, Stanford POS tagger, the rule-based bottom-up parser Wikipedia
Syntactic ambiguity	✗	✓	✓	✗	✓	✓
Lexical Ambiguity	✓	✗	✓	✓	✗	✓
Semantic Ambiguity	Scope	✓	✓	✓	✗	✓
Pragmatic Ambiguity	✗	✗	✗	✗	✗	✓
User interaction	High	Medium	Low	Low	Low	Medium

be collected from open-source SRS documents. Once data collection is complete, it will be analyzed using the developed prototype.

Evaluation is conducted to validate the research findings and to measure the performance and accuracy of the proposed approach. Evaluation is conducted in two steps, i.e., performance evaluation and output document verification.

3.3.1. Performance Evaluation. To evaluate the performance of the system, an evaluation methodology is used proposed by Hirschman and Thompson [46]. The performance evaluation methodology is based on three aspects, i.e., Criterion, Measure, and F-measure. To evaluate the results of the developed system, each element (Noun concepts, Verb concepts, and SBVR rules) of the system’s generated output was compared with the expert’s opinion (*Nexpert*) (sample solution). The element is classified as correct (*Ncorrect*), incorrect (*Nincorrect*), or missing (*Nmissing*).

3.3.2. The Output Document Verification. The resultant output document is stored in the XML format. To verify XML schema, the XML file is parsed to validation service W3C RDF/XML validation service [47]. If the document is correctly formatted, the W3C web service replicates the document into Triples and the graph format.

3.4. Research Findings and Conclusion. In the end, conclusion, scope, limitations, and further work improvements are being listed.

4. Design of the Proposed Approach

This section describes the design of the prototype tool of semiautomated NL-Requirements-SBVR-Rules transformation. The prototype comprises six basic stages, as shown in Figure 1.

4.1. Input Documents. The proposed approach takes two inputs:

- (a) An English text document (.txt file): the input is taken as a plain text file containing English written software requirements. It is assumed that the given English text is grammatically correct
- (b) Software Artifact: the system will accept a software artifact/model such as the UML class model (.ecore file) to validate the SBVR vocabulary
- (c) Wikipedia: the system will use Wikipedia’s assistance to associate meanings to the validated SBVR vocabulary.

4.2. Stage-1: Parse NL Text of Software Requirements. The parsing phase includes lexical and syntax analysis of the input software requirements. This phase involves processing units (ordered in a pipeline) to analyze complex English sentences. The parsing steps are as follows:

Lexical processing: this phase takes a plain text file containing an English SRS document as an input. Lexical processing has further subphases:

Tokenization: the first subphase of lexical processing is the tokenization of English text software requirements. The text is sliced into tokens. A single token is in fact a “sequence of characters” with collective meaning. During the tokenization phase, a sentence is sliced into token instances. Each such instance is known as lexemes.

Delimiters are used to identify lexemes: based on our requirement, each sentence of the English text is tokenized using StringTokenizer (str) and the output is stored as an array-list.

An example of input text: “A designer may work on many projects.” The generated tokenized output is: [A] [designer] [may] [work] [on] [many] [projects] [.]]. Such tokenized data will be used in syntactic analysis.

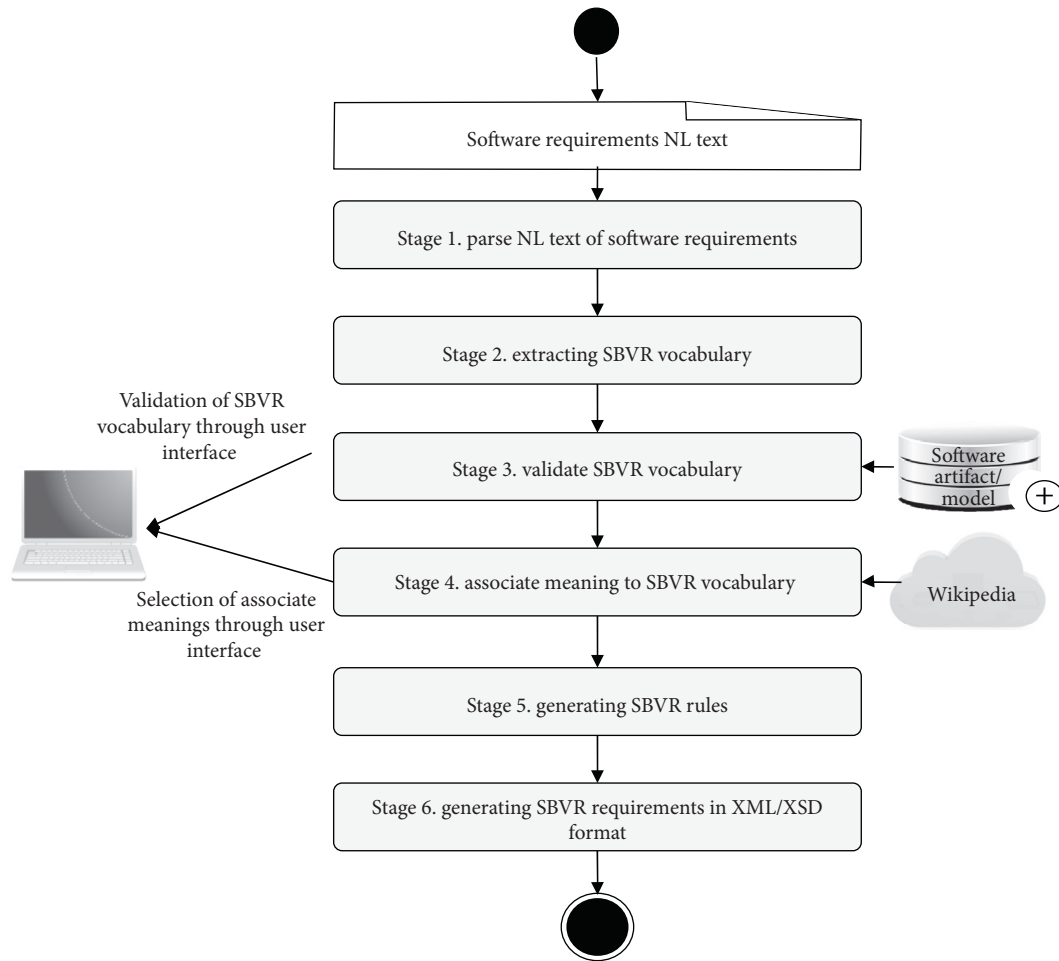


FIGURE 1: Algorithm of the proposed methodology.

Sentence splitting: the splitter spots the ends of a statement usually by a period “.”. It can also be used to split the sentence itself. We used Java Split() method to split the strings into substrings. Such substrings are stored in a String array. The Spit() method splits the string using the regular expression(RE) given inside the parameters while calling the method. Such split sentences will be used in later stages. Each identified sentence is stored separately in an array-list.

Parts-of-speech (POS) tagging: a POS Tagger reads text as an input in some language (in our case English language), tokenizes the input, and assigns POS tags to each token, such as nouns, adjectives, etc. In this subphase, basic POS tags are identified using Stanford POS tagger v3.0 [48]. The tagger is Java implemented software of the log-linear part-of-speech taggers. The overall accuracy of the Stanford POS tagger is 96.86% [48]. Penn Treebank is being used for tokenization. The tagger takes requirements as the input. The given input is transformed into tokens. After the completion of the tokenization stage, the tagger assigns POS tags to each identified token. The generated result is stored in an array-list for further processing.

An example of the tagger generated output:

[A/DT] [customer/NN] [can/MD] [have/VB] [two/CD] [Credit/NNP] [Cards/NNP] [./].

Syntactic processing: in this phase, the input is syntactically analyzed to produce POS tags. The above subphase is also performing syntactically analysis and generating POS tags using the Stanford POS tagger. The output generated by the Stanford POS tagger includes tagging, parse tree, and universal dependencies. Such output formats are not enough to perform semantic analysis and to generate the SBVR vocabulary and Rules. The SBVR generated rules have an English sentence format. Such format requires tags identification to generate an English sentence. The desired structure is

Subject + Verb + Object

Furthermore, the Stanford tokenizer is unable to identify action verbs (active voice and passive voice), models, and demonstratives. Stanford tokenizer is unable to convert English written quantifications to equivalent decimal numbers. Such identifications are crucial for SBVR vocabulary and Rules generation. To

generate an SBVR rule, we need to do a deep semantic analysis. Along with the Stanford POS tagger's generated tags, our proposed methodology requires additional information such as:

Logical formulation: logical formulations comprise logical operators such as AND, OR, NOT, implies, etc. Table 3 shows the possible tokens mapping to logical formulations:

Quantification: quantification specifies the scope of a subject. Possible quantifications can be identified from tokens by following mapping rules mentioned in Table 4:

Modal formulation: modal formulation depicts the relationship linking the meaning of another logic formulation and possible words or to acceptable words. Modal formulation specifies the seriousness of a constraint. Table 5 shows possible tokens mapping to the modal formulation:

To extract such information, based on methodology requirement, an enhanced version of a Stanford parser, the rule-based, bottom-up parser proposed by Bajwa et al. [49], is used to analyze input at the next level. The rule-based, bottom-up parser is based on English grammar rules. Such a parser takes tagged tokens as the input. Such tokens were generated in the previous subphase in Section 4.2. The enhanced version of the rule-based parser is capable of accepting more than one requirement at a time. Such a feature was not available in the original version. The parser overcomes the abovementioned limitations of the Stanford parser and will extract all necessary information required to perform semantic analysis. The generated output will be used to perform semantic analysis and to generate SBVR vocabulary and rules. The algorithm used by the rule-based parser to identify each tag/chunk is shown in Figure 2.

Sample input for the rule-based parser is [A/DT user/NN can/MD have/VB two/CD Debit/NNP Cards/NNP./.]

The generated output for the given input is shown in Figure 3. The generated output is saved in an array for further use.

Semantic Analysis: The phase identifies the meanings and inference of a certain string. A program is considered semantically reliable if all its variables, functions, classes, etc., must be appropriately defined; expressions and variables are following the model. This is a crucial phase of analysis as a semantically incorrect requirement will eventually produce an incorrect system. The proposed semantic analyzer analyzes the tags generated in syntactic processing and assigns corresponding roles to each tag. The process followed is shown in Figure 4.

The user uses this semantic table (see Table 6) to verify the roles of each token/chunk and hence the semantics of requirements. A single table can display the whole set of input requirements. Such identified roles assist in identifying the SBVR vocabulary in the later phase. All identified roles are stored in an array-list. Figure 5 exemplifies the output generated by the Semantic Parser.

TABLE 3: Tokens to logical formulations mapping.

Tokens	Logical formulation
"not," "no"	<i>negation</i> ($\neg a$)
"that," "and"	<i>conjunction</i> ($a \wedge b$)
"or"	<i>disjunction</i> ($a \vee b$)
"imply," "suggest," "if," "infer"	<i>implication</i> ($a \implies b$)

TABLE 4: Tokens to quantification mapping.

Tokens	Quantification
"more than," "greater than"	<i>at least</i> n
"less than"	<i>at most</i> n
token "equal to" or a positive statement	<i>exactly</i> n

TABLE 5: Tokens to quantification mapping.

Tokens	Modal formulation
Model verbs ("can," "may")	Structural requirement
Model verbs ("should," "must")	Behavioral requirement
Verb concept ("have to")	Behavioral requirement

4.3. Stage-2: Extracting the SBVR Vocabulary. This stage identifies primary SBVR vocabulary elements from the English input that was preprocessed in the previous stage 2. To write SBVR rules, we need to identify SBVR vocabulary elements. The steps to extract SBVR elements include

- (i) Extracting Unitary Noun Concept
- (ii) Extracting Individual Noun Concept
- (iii) Extracting Individual Verb Concepts
- (iv) Extracting Binary Verb Concepts
- (v) Extracting Characteristic/Unary Verb Concepts
- (vi) Extracting Unitary Verb Concepts
- (vii) Extracting Associative Verb Concept
- (viii) Extracting Partitive Verb Concept
- (ix) Extracting Quantification
- (x) Extracting Categorization

The final step is to create facts. An SBVR fact is a basic building block of the formal representation [38]. A fact type is based on identified verb concepts. A list of noun concepts and verb concepts is available in the SBVR vocabulary array-list. A fact type is generated by mapping such concepts. Atomic formalization is used to map the input requirement to an appropriate fact type. The generated list of the SBVR vocabulary consists of concepts and fact types. Such a vocabulary will be used as a reference throughout the generation of SBVR rules. A list of possibly extracted vocabulary elements from English text is shown in Table 7.

4.4. Stage-3: Validation of the SBVR Vocabulary. This phase validates that the elements of the SBVR vocabulary are consistent with the domain. The validation phase takes the list of the extracted SBVR vocabulary in the form of an array-list that comes from the previous phase and the second one is

- (1.1) Identify is, are, am, was, were as “subject in state”
- (1.2) Identify has, have, had as “subject in possession”
- (1.3) Identify EX (existential there) as “there”
- (1.4) Identify WDT (Wh-determiner) as “that”
- (1.5) Identify CD (cardinal number) and DT (determiner) as “quantifications”
 - (1.5.1) Convert english text number to decimal number (‘one’ and ‘a’ -> 1)
- (1.6) Identify CC (coordinating conjunction) as “conjunction”
 - (1.6.1) Convert to equivalent symbol (‘and’ -> \$\$, ‘or’ -> ||, ‘not’ -> !, ‘-?’ \$\$)
- (1.7) Identify IN and TO: (preposition or subordinating conjunction) as “preposition”
- (1.8) Identify NN (noun, singular or mass), NNP (proper noun, singular), NNPS (proper noun, plural), NNS (noun, plural), POS (possessive ending), and quantification
 - (1.8.1) Identify “subject” along with a conjunction
 - (1.8.2) Identify “object” along with conjunction and preposition
- (1.9) Identify “helping-verb” and “action-verb” using VB, MD, VBZ, VBD, VBN, VBDN, VBP

FIGURE 2: The rule-based parser algorithm.

tag	Chunk
asb	1 user
hvb	can
avb	have
ob	2 debit cards
./.	eos

FIGURE 3: An example of syntax analysis.

a software artifact/model such as a UML class model. To input such a model, the following two options can be used:

- (1) The user provides a UML class model of the respective business domain that will be used as a software artifact for the validation process.
- (2) The approach provides a repository of a large number of UML class models from various business domains. The user will perform a manual selection of relevant software artifacts/model from the available repository.
- (3) Once the selection of a UML class model has been made, the selected UML class model is parsed using the parseEcore parser to extract the metadata.
- (4) UML parser performs extraction on the resultant content. The extraction process includes the following steps:
 - (a) Extract classes
 - (b) Extract attributes
 - (c) Extract operations list
 - (d) Extract associations
 - (e) Extract generalization

Once the extraction process is complete, nodes are created and the parseEcoreparser represents the extracted features of the UML model in a hierarchical tree format, as shown in Figure 6.

The user will manually perform the mapping of extracted metadata to SBVR elements to validate the SBVR vocabulary, as shown in Table 8.

4.5. Stage-4: Associate Meanings. The key success of the SBVR is that it defines every vocabulary element; hence, eliminate ambiguity in terms and statements. This phase associates meanings to the validated SBVR vocabulary and eliminates “single word–multiple senses” ambiguity. Before the conversion of English text to SBVR rules, the meaning is associated with the input SBVR vocabulary to ensure that the resultant SBVR rules will be semantically associated with the relevant business domain. The system will look for associate meaning for each requirement in the Knowledge Base.

This phase receives two inputs; the SBVR vocabulary and Knowledge Base. Wikipedia is used as a knowledge base. Wikipedia is a reliable and general-purpose Knowledge Base accommodating all possible sets of meanings from different domains. Wikipedia is a fast and lightweight application; requires only an Internet connection; is easily accessible and has no specific memory requirement. A list of synonyms is also available on Wikipedia; hence, it eliminates “multiple words–one sense” ambiguity.

- (1) To associate meanings, the user has to select an SBVR vocabulary element.
- (2) The proposed system uses a Wikipedia parser to extract a list of possible meanings from Wikipedia. All possible scenarios for a selected SBVR vocabulary element will be displayed from Wikipedia using an interface, and the final selection will be left to the domain expert/analyst.
- (3) The user can select the associated meaning for all SBVR vocabulary elements.
- (4) The SBVR vocabulary is updated by adding associated meanings. Such associated meanings are added to the SBVR vocabulary for future assistance. The association of meanings will be of great assistance to the Analysis and Design team. There will be no ambiguity in the SBVR vocabulary in terms of
 - (i) Single word–multiple senses
 - (ii) Multiple words–one sense

Figure 7 shows added associate meanings in the SBVR vocabulary.

- (1) The input is an array from the previous syntactic processing phase.
- (2) Semantic analyzer tokenized the input using java stringtokenizer() method.
- (3) Each token/chunk (an english word) perform a specific role within a sentence such as “subject”, “object”, “adverb”, “preposition” etc. it is very important to identify such roles to understand the semantics of a requirement written as english text. The semantic analyzer identifies roles as:
- (4) Parser generates higher-order logic-based semantic representation. This will identify status of “subject” and “object” and label type as “state”, “possession” and “active”.
- (5) In the end parser performs atomic formulation. Atomic formulation includes the role binding for a particular role of the verb concept that is the basis of the atomic formulation. Atomic formulations are labeled as “is a”, “attribute of”, “akind of”, “belongs to”, “quantification” and “relation.”
- (6) End of each requirement is marked with “EOS” determiner. It helps to distinguish among different requirements.
- (7) The identified roles are displayed in the semantic table.

FIGURE 4: Steps followed for semantic analysis.

TABLE 6: An example of a semantic table.

Tag	Syntax	Type
Ssb	Subject	State
	OR	
Avb	Verb	
For, of, in, on, from, at, etc.	Preposition	
Integer	Quantification	
./.	Eos	

#	Chunk	Syntax	Quant	Logical	Type	Prep	EOS
1	Customer	Subject	1		Active		
2	Can	H.Verb					
3	Have	A.Verb					
4	Credit cards	Object	2				
5	Visa card	Object	1	AND			True

FIGURE 5: Algorithm for identifying subject part state sentence.

4.6. Stage-5: Generating SBVR Rules. Once the SBVR vocabulary is validated and associated with related meanings, the System is ready to generate SBVR rules. In this phase, SBVR rules are generated by using a rule-based parser. Such rules will be used to get the SBVR-based requirements specifications. Rule base parser contains a set of rules that maps SBVR elements with the SBVR vocabulary. The SBVR rule generator follows three steps to rule generation, which include extracting the SBVR Rule type, applying Semantic Formulation, and finally applying structured English notation.

Extracting the SBVR rule type: in this phase, each requirement is categorized either as a structural or a behavioral requirement. Such classification will be used to generate corresponding advice or behavioral rule. Following rules are applied to classify a requirement type.

Extracting advices: The requirements written in English language having auxiliary verbs, such as “can,” “may,” etc., are identified and classified as advice. For example, sentences representing state, e.g., “NBP is a bank,” or possession, e.g., “Bank cab has two cashiers,” can be

categorized as advice. Moreover, the English written requirements using general action verbs, such as consists, composed, equipped, etc., are also classified as structural requirements.

Extracting behavioral requirements: the English written requirements having auxiliary verbs such as “should,” “must” are identified and classified as a behavioral rule. Furthermore, the requirements having an action verb can be classified as a behavioral rule, e.g., “Cardholder provide a valid password.”

Applying semantic formulation: a set of such formulations are exercised to each fact type to generate an SBVR rule. The SBVR version 1.5 proposes two basic semantic formulations. These include Logical Formulation and Projections. Logic Formulation is further categorized as Atomic Formulations, Instantiation Formulations, Modal Formulations, Logical Operations, Quantifications, Objectification, Projecting Formulations, and Nominalizations of Propositions and Questions. Here, we are using the following three formulations concerning the context of the scope of the proposed research.

Logical formulation: an SBVR rule may consist of various Fact Types via logical operators such as AND, OR, NOT, implies, etc. Table 9 shows the possible tokens mapping to logical formulations:

Quantification: quantification specifies the scope of a concept. Possible quantifications can be identified from tokens by following mapping rules mentioned in Table 10:

Modal formulation: such formulation stipulates the weight of a constraint. Table 11 shows possible tokens mapping to the modal formulation:

The steps followed by the rule generator are as follows:

- (1) Identify new sentence
- (2) Identify numeric value
- (3) Identify each
- (4) Identify object types
- (5) Identify individual concept
- (6) Identify the verb concept

TABLE 7: Possible extracted vocabulary elements from English text.

Tags from English text requirement document	SBVR elements
Proper nouns	Individual concepts
Common nouns appearing in the subject part	Noun concepts or general concept
Common nouns appearing in the object part	Object type
Auxiliary and action verbs	Verb concepts
Auxiliary verbs and noun concepts	Fact types
Common nouns in the object part	Unary fact type: object type/individual concept without an action verb
Common nouns in the object part + auxiliary	Unary fact type with an action verb
Action verbs	binary fact type: object type/individual concept + verb + object type
Common nouns in the object part/proper nouns + auxiliary and action verbs + common nouns in the object part	Characteristic: Is-property-of fact type
Characteristic, adjectives or attributes, possessed nouns	Quantification with the respective noun concept
Indefinite articles, plural nouns, and cardinal numbers	Associative fact types
Associative or pragmatic relations	Partitive fact types
“Is-part-of,” or “included-in,” or “belong-to”	Categorization fact types
“is-category-of,” or “is-type-of,” “is-kind-of”	

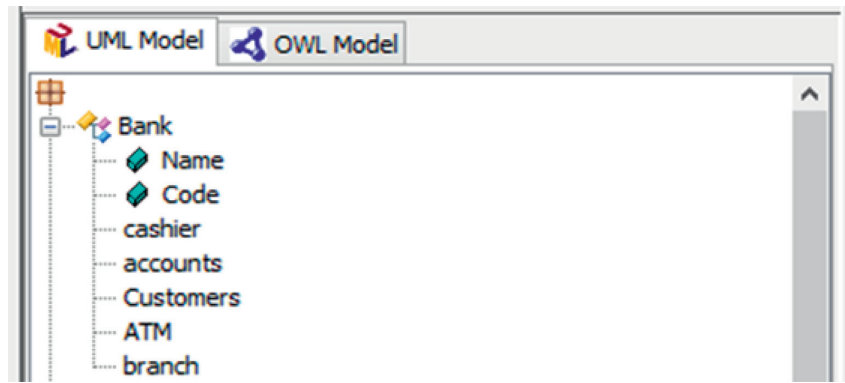


FIGURE 6: UML parser output.

TABLE 8: Equivalence between SBVR elements and UML model components.

SBVR elements	UML components
Individual concepts	Instances
Noun concepts or general concept	Classes
Object type	Classes
Verb concepts	Methods, operations
Fact types	Associations and generalizations
Unary fact type: object type/individual concept without an action verb	Attributes of a class
Unary fact type with an action verb	Unary relationship
binary fact type: object type/individual concept + verb + object type	Associations and generalizations
Characteristic: Is-property-of fact type	Attributes of class
Quantification with the respective noun concept	Multiplicity
Associative fact types	Caption of association
Partitive fact types	Generalizations
Categorization fact types	Aggregations

[Visa card](#)

- Concept type: [role](#)
- General concept type: [noun concept](#)
- Concept type: [individual concept](#)
- Logical term: [prepaid cards](#)
- Description: [\(pay from a cash account that has no checkwriting privileges\)](#)

FIGURE 7: An example of associated meaning to the SBVR vocabulary.

The algorithm followed to identify a new sentence is given in Figure 8.

4.6.1. *Applying the SBVR Notation.* The last phase is to apply an SBVR notation to generate SBVR-based requirements specifications. Such a format contains unambiguous and

TABLE 9: Tokens to logical formulations mapping.

Tokens	Logical formulation
“not,” “no”	<i>negation</i> ($\neg a$)
“that,” “and”	<i>conjunction</i> ($a \wedge b$)
“or”	<i>disjunction</i> ($a \vee b$)
“imply,” “suggest,” “if,” “infer”	<i>implication</i> ($a \implies b$)

TABLE 10: Tokens to quantification mapping.

Tokens	Quantification
“more than,” “greater than”	at least n
“less than”	at most n
token “equal to” or a positive statement	exactly n

TABLE 11: Tokens to modal formulation mapping.

Tokens	Modal formulation
Model verbs (“can,” “may”)	Structural requirement
Model verbs (“should,” “must”)	Behavioral requirement
Verb concept (“have to”)	Behavioral requirement

constant specifications. The end document will be an XML format file that uses the SBVR XMI XSD as its XML Schema. Such documents will be easy for a machine to process. The proposed approach supports SBVR Structured English. To apply Structured English:

- (i) The noun concepts are underlined, e.g., card
- (ii) The verb concepts are written as italic, e.g., *can*, *has*
- (iii) The keywords are bolded, i.e., SBVR keywords, e.g., **each**, **at least**, **at most**, **obligatory**, etc.
- (iv) The individual concepts are double-underlined, e.g., black cat

For example: “A person’s nationality should be British.” will be translated as per SBVR rule as: “It is obligatory that each a person’s nationality should be British.”

4.7. Stage-6: The Output. The output is saved and exported in an XML format. The file contains the SBVR vocabulary and SBVR rules. To verify the XML schema, the XML file can be parsed to any validation service such as W3C RDF Validation Service [47]. Such a file is an interchangeable, platform-independent, easy machine process document, providing a clear, unambiguous, consistent, and complete SRS document.

5. Implementation of the Approach

To better understand the stages, let us observe the interaction of the components during their defined scenarios of NL to SBVR Rules transformation.

- (i) The system takes two inputs—user requirements and the UML model for validation. Requirements are written by the user either in a text file (A1 from Figure 9) or directly on the SBVR editor pane (A2 from Figure 9). Such requirements are used to

extract the SBVR vocabulary. The evaluations of the proposed approach use a text file to input requirements. The UML model was selected from the available UML model repository. The user selects a domain-specific UML model to perform mapping from the UML model to SBVR vocabulary set.

- (ii) As the “Generate SBVR” button is clicked, the Rule-Based Parser [24] extracts the SBVR vocabulary from the NL text software requirements. The Parser identifies the concepts along with the concept type and its general concept (A3 from Figure 9). Along with the concepts, the parser also identifies the fact type of related concepts.
- (iii) The UML model is used to validate the generated SBVR vocabulary. This validation procedure was performed manually. The user mapped the identified SBVR vocabulary elements to the UML model items.
- (iv) The user can add semantics using the Wikipedia parser. The Wikipedia parser extracts the list of meanings from the Internet related to the SBVR vocabulary (A4 from Figure 9). The user can select the domain-specific meaning (A5 from Figure 9). Such a selected meaning is associated with the corresponding vocabulary concept (A6 from Figure 9).
- (v) The Rule-Based Parser generates the SBVR Rules. Such SBVR rules include Structural rules and Behavioral rules (A7 from Figure 9). Once the SBVR rules are generated, the SBVR Structured Notation is applied to the rules. The SBVR notation makes the rules more readable.

XML Parser generates a RDF/XML schema (A8 from Figure 9). Such a file provides a consistent, interchangeable, platform-independent schema. The generated RDF schema is also validated through the RDF validator available at <http://www.w3.org>. An example is shown in Figures 10 and 11.

6. Results

The proposed approach of NL to SBVR Rules extraction was evaluated using seven sets of requirements (T_1, T_2, \dots, T_7). Each set consists of 50 randomly selected requirements. The selected requirements were syntactically valid. The requirement set was parsed to prototype. A sample of extracted Noun concepts, Verb concepts, Fact types, and generated SBVR rules are presented in Table 12. The sampling requirement set T is a subset of the requirements set T_1 . T_1 is related to the domain “car rental services.” For 5 sample input sentences of the requirement set T , our designed prototype has extracted 8 Noun Concepts and 5 Verb Concepts. These Noun Concepts and Verb Concepts are processed to create 5 Fact types. Once the necessary extraction is performed, Semantic formulation and SBVR notation are applied on Fact type to generate SBVR Rules.

- (1) Identify new sentence
 - (1.1) Identify the negative sentence
 - (1.1.1) Mark sentence as “negative”
 - (1.2) Identify POS tag as “JJR” and “than” in the sentence
 - (1.2.1) Identify “more”, “greater” and “most”
 - (1.2.1.1) If sentence is negative: add “at most” in rule
 - (1.2.1.2) Else add “at least” in rule
 - (1.2.2) Identify “less”, “smaller”, “least”
 - (1.2.2.1) If sentence is negative: add “at least” in rule
 - (1.2.2.2) Else add “at most” in rule
 - (1.3) Identify “maximum” in sentence
 - (1.3.1) If sentence is negative: add “at most” in rule
 - (1.3.2) Else add “at least” in rule
 - (1.4) Identify “minimum” in sentence
 - (1.4.1) If sentence is negative: add “at least” in rule
 - (1.4.2) Else add “at most” in rule
 - (1.5) Identify “equal to” in sentence
 - (1.5.1) If sentence is negative: add “not exactly” in rule
 - (1.5.2) Else add “exactly” in rule
 - (1.6) Identify “PRP” in POS tags
 - (1.6.1) Add corresponding vocabulary element in rule
 - (1.7) Identify “exactly” in sentence
 - (1.7.1) If sentence is negative: add “not exactly” in rule
 - (1.7.2) Else add “exactly” in rule
 - (1.8) Identify “RB” in POS tags
 - (1.8.1) If corresponding vocabulary element belongs to {more, greater, less, exactly, least, most} then add new sentence
 - (1.8.2) Else add corresponding vocabulary element in rule
- (2) End

FIGURE 8: SBVR rule generation.

Table 13 actually specifies the extracted elements for each requirement set (T_1, T_2, \dots, T_7) used for the construction of SBVR rules. First of all, using the prototype, we identify Noun Concepts and Verb Concepts. Using these concepts, the prototype generates Fact types. Such fact types are further processed to generate SBVR rules.

The Requirement set T_1 is processed by the prototype and 89 Noun Concepts and 48 Verb Concepts are extracted. These concepts are processed to construct 50 Fact types. The prototype finally generates 50 Rules. The prototype has extracted a total of 237 elements for T_1 . The process is followed for each requirement set T_1, T_2, \dots, T_7 . A sample of such extractions for rule generation is depicted in Table 12.

After seven iterations of seven case studies, the total number of extracted elements such as Noun Concept are 624 and Verb Concept are 332. The prototype has constructed the 346 Fact Type. The prototype has successfully generated 350 Rules for seven case studies consisting of 350 requirements collectively.

7. Analysis and Discussion

To evaluate the results of the approach, we prepared a sample test case requirements set (T_1, T_2, \dots, T_7). An expert manually evaluated the requirements sets to create sample data N_{expert} , as shown in Table 14. N_{expert} comprises extracted Noun Concepts, Verb Concepts, constructed fact type, and generated Rules. The purpose is to validate the prototype-generated output. The sum of the output generated by the prototype is labeled as N_{total} . N_{correct} is the

element correctly identified by the prototype. $N_{\text{incorrect}}$ is the element that is identified by the prototype but is incorrect when compared with N_{expert} . N_{missing} is the element that the prototype is unable to identify or process. A comparison is performed in Table 14 between the extracted concepts, constructed fact types, generated Rules by the prototype, and with manually evaluated requirements; N_{expert} .

The expert (N_{expert}) has identified 94 Noun Concepts and 50 Verb Concepts from requirements set T_1 . The expert (N_{expert}) then constructed 50 fact types based on extracted Noun and Verb Concepts. In the end, the expert (N_{expert}) has generated 50 SBVR Rules. In comparison, the prototype has identified 89 Noun Concepts out of which 86 are correct (N_{correct}), 3 are incorrect ($N_{\text{incorrect}}$), and 5 are missing (N_{missing}). The prototype then extracted 48 (N_{total}) Verb Concepts, from which 46 are correct (N_{correct}) and 2 are incorrect ($N_{\text{incorrect}}$). Two Verb Concepts are missing in the list. Afterward, the prototype has constructed 50 Fact types (N_{total}). Two Fact types are incorrect ($N_{\text{incorrect}}$). In the end, the SBVR Rules are constructed using Fact types. A total of 237 elements are identified out of which 230 are correctly identified, 7 are incorrect identifications, while 7 elements are missing. The process is repeated for each requirement set (T_1, T_2, \dots, T_7), as shown in Table 14.

The expert successfully extracted 1752 elements for seven case studies. According to the used evaluation methodology, Table 14 shows 1652 identified elements, of which 1595 are correct, 57 are incorrect, and 100 are missing SBVR elements. For each requirement set, the rules generation rate is 100%. The Fact Type identification results are also very satisfactory. Out of 350 total

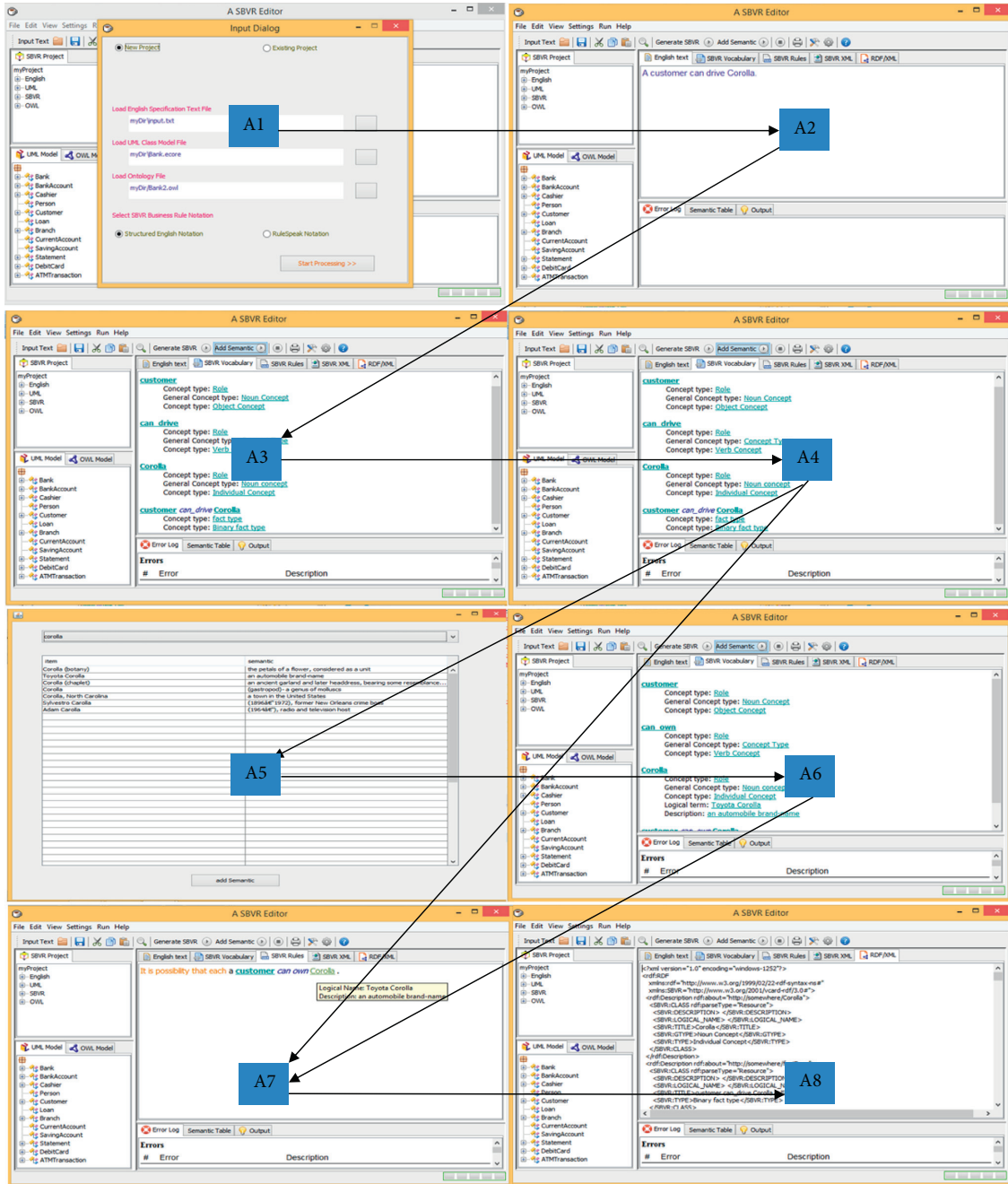


FIGURE 9: Interaction of different components implementing the NL to SBVR rules transformation.

requirements, 339 fact Types are correctly identified. The approach has incorrectly tagged some Noun Concepts as Verb Concepts and vice versa. The reason is the lack of a strong dataset. For example, in set T1, 3 Noun Concepts and 2 Verb Concepts are incorrectly tagged. However, overall initial results are very encouraging. Prototype performance is measured using three metrics: Recall, Precision, and *F*-value. Such metrics are widely used to assess NL-based data extraction systems. Such metrics help to have a comparison of system predictions versus actual values. The Recall is a measure of the correctly identified elements by the system.

$$R = \frac{N_{\text{system}}}{N_{\text{expert}}}, \tag{1}$$

where N_{system} is the number of the system's generated correct results, and N_{expert} is the number of human expert's generated sample results.

The precision is a ratio between correct and incorrect elements identified by the system.

$$P = \frac{N_{\text{correct}}}{N_{\text{incorrect}} + N_{\text{correct}}}, \tag{2}$$

Number	Subject	Predicate	Object
1	http://somewhere/factType	http://www.w3.org/2001/vcard-rdf/3.0#CLASS	genid:A78
2	genid:A78	http://www.w3.org/2001/vcard-rdf/3.0#DESCRIPTION	""
3	genid:A78	http://www.w3.org/2001/vcard-rdf/3.0#LOGICAL_NAME	""
4	genid:A78	http://www.w3.org/2001/vcard-rdf/3.0#TITLE	"customer can have Visa_Card"
5	genid:A78	http://www.w3.org/2001/vcard-rdf/3.0#TYPE	"Binary fact type"
6	http://somewhere/SbvrRule	http://www.w3.org/2001/vcard-rdf/3.0#CLASS	genid:A79
7	genid:A79	http://www.w3.org/2001/vcard-rdf/3.0#DESCRIPTION	""
8	genid:A79	http://www.w3.org/2001/vcard-rdf/3.0#LOGICAL_NAME	""
9	genid:A79	http://www.w3.org/2001/vcard-rdf/3.0#TITLE	"It is possibility that each a customer can have one Visa_Card"
10	genid:A79	http://www.w3.org/2001/vcard-rdf/3.0#TYPE	"SbvrRule"
11	http://somewhere/customer	http://www.w3.org/2001/vcard-rdf/3.0#CLASS	genid:A80
12	genid:A80	http://www.w3.org/2001/vcard-rdf/3.0#DESCRIPTION	""
13	genid:A80	http://www.w3.org/2001/vcard-rdf/3.0#LOGICAL_NAME	""
14	genid:A80	http://www.w3.org/2001/vcard-rdf/3.0#TITLE	"customer"
15	genid:A80	http://www.w3.org/2001/vcard-rdf/3.0#TYPE	"Noun Concept"

FIGURE 10: W3C generated triplets verifying system generated RDF/XML output.

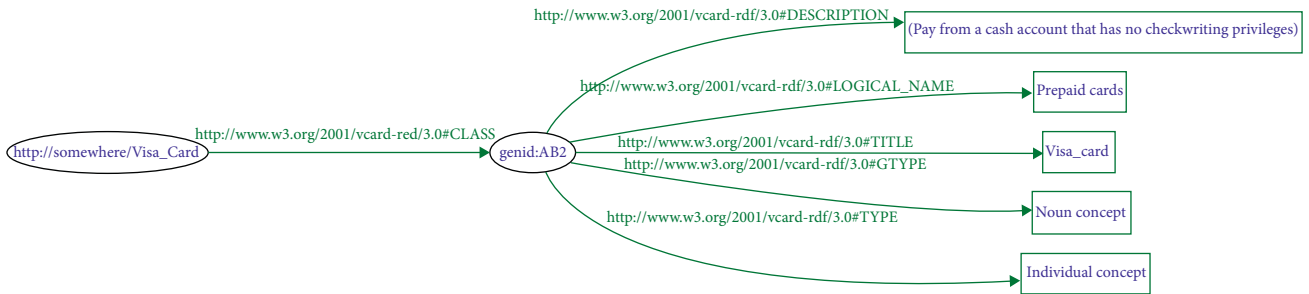


FIGURE 11: W3C-generated graphs verifying system-generated RDF/XML output.

TABLE 12: Result of executed NL text software requirements.

Requirements set	Result type	Extracted result
T	SBVR Noun Concept	<u>company_name</u> , <u>Rent-a-Car</u> , <u>company</u> , <u>drivers</u> , <u>hours</u> , <u>day</u> , <u>cars</u> , <u>Ahmad</u>
	SBVR Verb Concept	<u>is</u> , <u>employed</u> , <u>work</u> , <u>owns</u>
T	SBVR Fact Type	<u>company_name</u> <i>is</i> <u>Rent-a-Car</u> <u>company</u> <i>employed</i> <u>drivers</u> <u>drivers</u> <i>work</i> <u>hours</u> <u>company</u> <i>owns</i> <u>cars</u> <u>Ahmad</u> <i>is</i> <u>driver</u>
	SBVR Rules	It is permitted that each <u>company_name</u> <i>is</i> "Rent-a-Car," each <u>company</u> <i>employed</i> 5 <u>driver</u> .each <u>driver</u> <i>work</i> 8 <u>hours</u> each a <u>day</u> .each <u>company</u> <i>owns</i> 5 <u>car</u> . It is permitted that <u>Ahmad</u> <i>is</i> each <u>name</u> of each a <u>driver</u> .

TABLE 13: Extracted elements by prototype.

Type/Metrics	Requirements set								Total
	T1	T2	T3	T4	T5	T6	T7		
Noun Concept	89	93	92	90	87	92	81	624	
Verb Concept	48	48	47	47	50	46	46	332	
Fact Type	50	48	49	50	50	49	50	346	
SBVR Rules	50'	50	50	50	50	50	50	350	

TABLE 14: Comparison of sample and prototype generated results.

Requirements set	Type/metrics	N_{expert}	N_{total}	N_{correct}	$N_{\text{incorrect}}$	N_{missing}
T1 (50 requirements)	Noun Concept	94	89	86	3	5
	Verb Concept	50	48	46	2	2
	Fact Type	50	50	48	2	0
	Rules	50	50	50	0	0
T2 (50 requirements)	Noun Concept	105	93	93	0	12
	Verb Concept	50	48	45	3	2
	Fact Type	50	48	47	1	2
	Rules	50	50	50	0	0
T3 (50 requirements)	Noun Concept	100	92	86	6	8
	Verb Concept	50	47	44	3	3
	Fact Type	50	49	49	0	1
	Rules	50	50	50	0	0
T4 (50 requirements)	Noun Concept	99	90	86	4	9
	Verb Concept	50	47	45	2	3
	Fact Type	50	50	47	3	0
	Rules	50	50	50	0	0
T5 (50 requirements)	Noun Concept	100	87	82	5	13
	Verb Concept	50	50	49	1	0
	Fact Type	50	50	49	1	0
	Rules	50	50	50	0	0
T6 (50 requirements)	Noun Concept	102	92	87	5	10
	Verb Concept	52	46	42	4	6
	Fact Type	50	49	45	4	1
	Rules	50	50	50	0	0
T7 (50 requirements)	Noun Concept	100	81	75	6	19
	Verb Concept	50	46	44	2	4
	Fact Type	50	50	50	0	0
	Rules	50	50	50	0	0
Total Identified Elements		1752	1652	1595	57	100

where N_{correct} is the number of correct results, and $N_{\text{incorrect}}$ is the number of incorrect results generated by the developed system.

F -measure is the measure of the prototype's accuracy.

$$F = \frac{2(P)(R)}{P + R}, \quad (3)$$

where P is the precision value and R is the recall value.

Table 15 describes the calculated recall, precision, and F -value of the prototype for each NL requirements set ($T1$, $T2$, ..., $T7$). The calculation is based on the abovementioned Equations (1), (2), and (3). Recall, Precision, and F -value are separately calculated for each set of requirements ($T1$, $T2$, ..., $T7$). After that, an average value is calculated. $T1$ has the highest Recall (0.97) and F -value (0.97), while $T2$ has the highest Precision (0.98). $T7$ has the lowest Recall (0.91) value, while $T6$ has the lowest Precision (0.95). The reason for such low values is a weak dataset. The average recall for the SBVR SRS document is calculated as 0.94, while the average precision is calculated as 0.97. The average F -value is computed as 0.95. The results of this initial performance evaluation are very encouraging. The results support both the used approach and the potential of this tool in general. Figure 12 shows the graphical representation of the tool's evaluation.

Figure 13 displays the N_{sample} , N_{correct} , $N_{\text{incorrect}}$, and N_{missing} evaluation results of the proposed tool for the seven

case studies, $T1$ – $T7$. The blue column shows results for N_{sample} , the red column shows the result for N_{correct} , the green column shows results for $N_{\text{incorrect}}$, while the purple column shows the results for N_{missing} . $T2$ has the highest value of N_{correct} elements, while $T6$ has the lowest value of N_{correct} elements. $T1$ has the lowest value of N_{missing} elements, while $T7$ has the highest value of N_{missing} elements.

The abovementioned Figure 13 graphically represents the results of Recall, precision, and F -value generated by the prototype while processing seven different case studies ($T1$, $T2$, ..., $T7$). According to our calculated results, $T1$ has high Recall, $T2$ has high precision, and $T1$ has high F -Value. In contrast, $T7$ has the lowest Recall value, $T6$ has the lowest Precision, and $T7$ has the lowest F -measure value. The reason for such low values is a weak dataset.

The resultant output is stored in RDF/XML format. To verify XML schema, the XML file is parsed by an online validation service named W3C RDF Validation Service [47]. This web service successfully replicates the output into triples and graphs. A sample section of web-generated output is shown in Figure 10.

The results presented above show that it is convenient and time-saving to formulate a semantically formal and controlled illustration using our proposed approach. Figure 10 shows the validation results and Figure 11 shows the W3C-generated graphs verifying system-generated RDF/XML output.

TABLE 15: Evaluation results of prototype.

Input	N_{expert}	N_{correct}	$N_{\text{incorrect}}$	N_{missing}	Recall	Precision	F -value
T1	244	230	7	7	0.97	0.97	0.97
T2	255	235	4	16	0.94	0.98	0.96
T3	250	229	9	12	0.95	0.96	0.96
T4	249	228	9	12	0.95	0.96	0.96
T5	250	230	7	13	0.95	0.97	0.96
T6	254	224	13	17	0.93	0.95	0.94
T7	250	219	8	23	0.91	0.96	0.94
Total	1752	1595	57	100			
Average					0.94	0.97	0.95

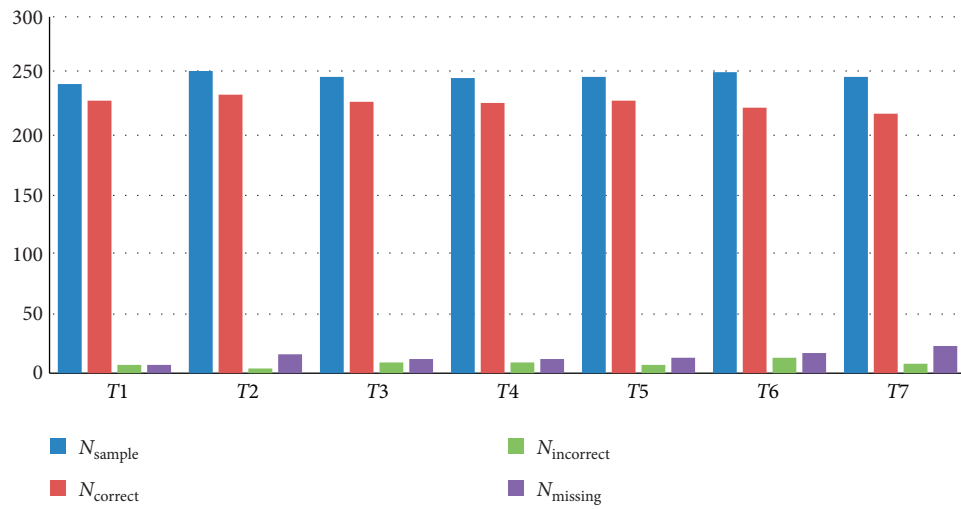


FIGURE 12: Evaluation result of the proposed tool.

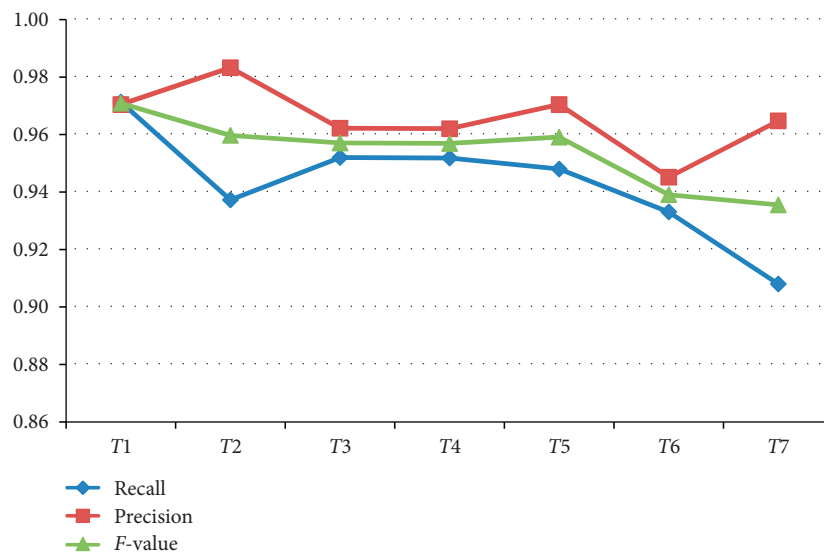


FIGURE 13: Recall, precision, and F -values of the evaluation result.

8. Scope and Limitations

We recognize a set of limits of this investigative study, which would permit the researchers to present a more suitable design for the succeeding research phase. Our study used only one expert to create sample data, which is not a representative set of the population of experts. Furthermore, the experience level of the expert can also put limitations on the credibility of the sample data.

It is assumed that the given English text to the prototype as the input is grammatically correct. The dataset used for experimental evaluation contains complete sentences, i.e., subject–verb–object. Results may be different for incomplete or incorrect sentences.

9. Conclusion and Future Work

The major goal of this research work was to automate the practice of software requirement elicitation and requirement specification while tackling the ambiguous temperament of NL such as English. At the same time, we aim to generate a controlled representation of such requirements, which is domain-independent and acceptable for the business industry. To tackle the challenge, we have proposed an NL-based approach. We developed a prototype based on the proposed approach. The prototype parse English written software requirement and generate the SBVR-based controlled representation. The prototype further extracts the SBVR vocabulary to generate SBVR rules. The proposed prototype performs the lexical, syntactic, and semantic analyzes concerning SBVR rules. While extracting the SBVR vocabulary, the assignment of meanings to each vocabulary element removes the chances of ambiguity. The prototype generates rules written in the SBVR-based controlled language. Such rules are clear and unambiguous among the business community. The output of the whole process is stored in XML format, which is portable and platform-independent. Also, the generated output document is easily convertible into any other format for further processing. Such documents will be easy for the machine to process.

We have, successfully, evaluated the proposed approach on seven case studies with the aid of a developed prototype to support our proposed approach. Our proposed prototype can be used for automated Object-oriented analysis and design (OOA&D) of NL-software requirements. In addition, the prototype offers a higher accuracy as compared to other available NL-based tools. As shown in the Results section, the recall value of 0.94 and precision value of 0.97 results obtained from seven case studies for software requirements by using our prototype are very encouraging. Likewise, the resultant F-value of 0.95 is also satisfactory. Hence, the results of our assessment show the encouraging performance of our developed tool in terms of usability, time, and accuracy.

Beforehand, research has been carried out in large amount for the automation of SRS document using NLP-based approaches, but comparatively, little effort has been done on the approaches based on CNL representing requirement specification. For that reason, countless aspects need to be investigated while using the SBVR-based controlled representation of requirements specification. The

future work is to validate the extracted vocabulary automatically using UML and ontology models. Automated validation of such data can be helpful in automated conceptual modeling of the NL SRS document.

Data Availability

Data are available upon request to the corresponding author.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

- [1] A. O. J. A. Sabriye and W. M. N. W. Zainon, "A framework for detecting ambiguity in software requirement specification," in *Proceedings of the 2017 8th International Conference on Information Technology (ICIT)*, pp. 209–213, IEEE, Amman, Jordan, May 2017.
- [2] T. Hovorushchenko and O. Pomorova, "Methodology of evaluating the sufficiency of information on quality in the software requirements specifications," in *Proceedings of the 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, pp. 370–374, IEEE, Kyiv, Ukraine, May 2018.
- [3] M. Luisa, F. Mariangela, and N. I. Pierluigi, "Market research for requirements analysis using linguistic tools," *Requirements Engineering*, vol. 9, no. 1, pp. 40–56, 2004.
- [4] E. Kamsties and B. Peach, "Taming ambiguity in natural language requirements," in *Proceedings of the Thirteenth International Conference on Software and Systems Engineering and Applications*, Paris, France, December 2000.
- [5] M. P. S. Bhatia, A. Kumar, and R. Beniwal, "Ontology based framework for detecting ambiguities in software requirements specification," in *Proceedings of the 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 3572–3575, IEEE, New Delhi, India, March 2016.
- [6] M. Bano, "Addressing the challenges of requirements ambiguity: a review of empirical literature," in *Proceedings of the 2015 IEEE Fifth International Workshop on Empirical Requirements Engineering (EmpIRE)*, pp. 21–24, IEEE, Ottawa, Canada, August 2015.
- [7] D. M. Berry, E. Kamsties, and M. M. Krieger, *From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity*, University of Waterloo, Waterloo, Canada, 2003, <https://cs.uwaterloo.ca/~dberry/handbook/ambiguityHandbook.pdf>.
- [8] V. Basili, G. Caldiera, F. Lanubile, and F. Shull, "Studies on reading techniques," in *Proceedings of the Twenty-First Annual Software Engineering Workshop*, vol. 96, p. 002, Greenbelt, MD, USA, December 1996.
- [9] E. Kamsties, "Understanding ambiguity in requirements engineering," in *Engineering and Managing Software Requirements*, pp. 245–266, Springer, Berlin, Germany, 2005.
- [10] S. McConnell, *Code Complete*, Pearson Education, London, UK, 2004.
- [11] A. Umer and I. S. Bajwa, "Minimizing ambiguity in natural language software requirements specification," in *Proceedings of the IEEE Sixth International Conference on Digital Information Management (ICDIM 2011)*, pp. 102–107, Melbourne, Australia, 2011.
- [12] A. Takoshima and M. Aoyama, "Assessing the quality of software requirements specifications for automotive software

- systems,” in *Proceedings of the 2015 Asia-Pacific Software Engineering Conference (APSEC)*, pp. 393–400, IEEE, New Delhi, India, December 2015.
- [13] F. Zait and N. Zarour, “Addressing lexical and semantic ambiguity in natural language requirements,” in *Proceedings of the 2018 Fifth International Symposium on Innovation in Information and Communication Technology (ISIICT)*, pp. 1–7, IEEE, Amman, Jordan, November 2018.
- [14] A. Chikh and H. Alajmi, “Towards a dynamic software requirements specification,” in *Proceedings of the 2014 World Congress on Computer Applications and Information Systems (WCCAIS)*, pp. 1–7, IEEE, Hammamet, Tunisia, January 2014.
- [15] R. Schwiter, “Controlled natural languages for knowledge representation,” in *Proceedings of the Coling 2010: Posters*, pp. 1113–1121, Beijing, China, August 2010.
- [16] R. Denaux, V. Dimitrova, A. G. Cohn, C. Dolbear, and G. Hart, “Rabbit to OWL: ontology authoring with a CNL-based tool,” in *Proceedings of the International Workshop on Controlled Natural Language*, pp. 246–264, Springer, Marzetto, Italy, June 2009.
- [17] C. White and R. Schwiter, “An update on PENG light,” in *Proceedings of the Australasian Language Technology Association Workshop 2009*, pp. 80–88, Sydney, Australia, December 2009.
- [18] G. Sandhu and S. Sikka, “State-of-art practices to detect inconsistencies and ambiguities from software requirements,” in *Proceedings of the International Conference on Computing, Communication & Automation*, pp. 812–817, IEEE, Greater Noida, India, May 2015.
- [19] P. Martin, “Knowledge representation in CGLF, CGIF, KIF, frame-CG and formalized-English,” in *Proceedings of the International Conference on Conceptual Structures*, pp. 77–91, Springer, Borovets, Bulgaria, July 2002.
- [20] U. S. Shah and D. C. Jinwala, “Resolving ambiguities in natural language software requirements: a comprehensive survey,” *ACM SIGSOFT Software Engineering Notes*, vol. 40, no. 5, pp. 1–7, 2015.
- [21] F. Friedrich, J. Mendling, and F. Puhlmann, “Process model generation from natural language text,” in *Proceedings of the International Conference on Advanced Information Systems Engineering*, pp. 482–496, Springer, London, UK, June 2011.
- [22] H. Afreen and I. S. Bajwa, “Generating UML class models from SBVR software requirements specifications,” in *Proceedings of the 23rd Benelux Conference on Artificial Intelligence (BNAIC 2011)*, pp. 23–32, Ghent, Belgium, 2011.
- [23] S. J. Korner and T. Brumm, “RESI-a natural language specification improver,” in *Proceedings of the 2009 IEEE International Conference on Semantic Computing*, pp. 1–8, IEEE, Berkeley, CA, USA, September 2009.
- [24] L. Mich and R. Garigliano, “Ambiguity measures in requirement engineering,” in *Proceedings of the International Conference on Software Theory and Practice*, ICS, Beijing, China, August 2000.
- [25] S. F. Tjong, “Avoiding ambiguity in requirements specifications,” Ph.D. thesis, https://cs.uwaterloo.ca/~dberry/FTP_SITE/tech.reports/TjongThesis.pdf, University of Nottingham, Nottingham, UK, 2008.
- [26] H. Yang, A. De Roeck, V. Gervasi, A. Willis, and B. Nuseibeh, “Analysing anaphoric ambiguity in natural language requirements,” *Requirements Engineering*, vol. 16, no. 3, p. 163, 2011.
- [27] E. Kamsties, D. M. Berry, B. Paech, E. Kamsties, D. M. Berry, and B. Paech, “Detecting ambiguities in requirements documents using inspections,” in *Proceedings of the First Workshop on Inspection in Software Engineering (WISE’01)*, pp. 68–80, Paris, France, July 2001.
- [28] A. Ferrari, G. Lipari, S. Gnesi, and G. O. Spagnolo, “Pragmatic ambiguity detection in natural language requirements,” in *Proceedings of the 2014 IEEE 1st International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)*, pp. 1–8, IEEE, Karlskrona, Sweden, August 2014.
- [29] H. Kaiya and M. Saeki, “Ontology based requirements analysis: lightweight semantic processing approach,” in *Proceedings of the Fifth International Conference on Quality Software (QSIC’05)*, pp. 223–230, IEEE, Melbourne, Australia, September 2005.
- [30] B. Gleich, O. Creighton, and L. Kof, “Ambiguity detection: towards a tool explaining ambiguity sources,” in *Proceedings of the International Working Conference on Requirements Engineering: Foundation for Software Quality*, pp. 218–232, Springer, Essen, Germany, June 2010.
- [31] O. Al-Harbi, S. Jusoh, and N. Norwawi, “Handling ambiguity problems of natural language interface for question answering,” *International Journal of Computer Science Issues (IJCSI)*, vol. 9, no. 3, p. 17, 2012.
- [32] R. P. Verma and M. R. Beg, “Representation of knowledge from software requirements expressed in Natural Language,” in *Proceedings of the 2013 6th International Conference on Emerging Trends in Engineering and Technology*, pp. 154–158, IEEE, Nagpur, India, December 2013.
- [33] K. D. Gill, A. Raza, A. M. Zaidi, and M. M. Kiani, “Semi-automation for ambiguity resolution in open source software requirements,” in *Proceedings of the 2014 IEEE 27th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 1–6, IEEE, Toronto, Canada, May 2014.
- [34] A. K. Massey, R. L. Rutledge, A. I. Antón, and P. P. Swire, “Identifying and classifying ambiguity for regulatory requirements,” in *Proceedings of the 2014 IEEE 22nd International Requirements Engineering Conference (RE)*, pp. 83–92, IEEE, Karlskrona, Sweden, August 2014.
- [35] S. W. Ali, Q. A. Ahmed, and I. Shafi, “Process to enhance the quality of software requirement specification document,” in *Proceedings of the 2018 International Conference on Engineering and Emerging Technologies (ICEET)*, pp. 1–7, IEEE, Lahore, Pakistan, February 2018.
- [36] D. Popescu, S. Rugaber, N. Medvidovic, and D. M. Berry, “Reducing ambiguities in requirements specifications via automatically created object-oriented models,” in *Proceedings of the Monterey Workshop*, pp. 103–124, Springer, Monterey, CA, USA, September 2007.
- [37] N. E. Fuchs, K. Kaljurand, and T. Kuhn, “Attempto controlled English for knowledge representation,” in *Reasoning Web*, pp. 104–124, Springer, Berlin, Heidelberg, 2008.
- [38] <http://www.omg.org/spec/SBVR/1.5/PDF>.
- [39] I. S. Bajwa, M. G. Lee, and B. Bordbar, “SBVR business rules generation from natural language specification,” in *Proceedings of the 2011 AAAI Spring Symposium Series*, San Francisco, CA, USA, March 2011.
- [40] S. Ramzan, I. S. Bajwa, I. U. Haq, and M. A. Naeem, “A model transformation from NL to SBVR,” in *Proceedings of the Ninth International Conference on Digital Information Management (ICDIM 2014)*, pp. 220–225, IEEE, Phitsanulok, Thailand, September 2014.
- [41] P. Danenas, T. Skersys, and R. Butleris, “Natural language processing-enhanced extraction of SBVR business vocabularies and business rules from UML use case diagrams,” *Data & Knowledge Engineering*, vol. 128, Article ID 101822, 2020.

- [42] P. B. F. Njonko and W. El Abed, "From natural language business requirements to executable models via SBVR," in *Proceedings of the 2012 International Conference on Systems and Informatics (ICSAI2012)*, pp. 2453–2457, IEEE, Yantai, China, May 2012.
- [43] F. L. Siqueira, T. C. de Sousa, and P. S. M. Silva, "Using BDD and SBVR to refine business goals into an Event-B model: a research idea," in *Proceedings of the 2017 IEEE/ACM 5th International FME Workshop on Formal Methods in Software Engineering (FormaliSE)*, pp. 31–36, IEEE, Buenos Aires, Argentina, May 2017.
- [44] P. K. Chittimalli and K. Anand, "Domain-independent method of detecting inconsistencies in sbvr-based business rules," in *Proceedings of the International Workshop on Formal Methods for Analysis of Business Systems*, pp. 9–16, Singapore, September 2016.
- [45] P. Arnold and E. Rahm, "Automatic extraction of semantic relations from wikipedia," *International Journal on Artificial Intelligence Tools*, vol. 24, no. 2, Article ID 1540010, 2015.
- [46] L. Hirschman and H. S. Thompson, "Chapter 13 evaluation: overview of evaluation in speech and natural language processing," in *Survey of the State of the Art in Human Language Technology*, R. A. Cole, H. Mariani, J. Uszkoreit et al., Eds., pp. 114–126, Cambridge University Press, Cambridge, UK, 1995, <http://cslu.cse.ogi.edu/HLTSurvey/HLTSurvey.html>.
- [47] <https://www.w3.org/RDF/Validator/rdfval>, 2020.
- [48] K. Toutanova and C. Manning, "Enriching the knowledge sources used in a maximum entropy part-of-speech tagger," in *Proceedings of the 2000 Joint SIGDAT Conference EMNLP/VLC*, vol. 63–71, Hong Kong, China, 2000.
- [49] I. S. Bajwa, A. Samad, and S. Mumtaz, "Object oriented software modeling using NLP based knowledge extraction," *European Journal of Scientific Research*, vol. 35, no. 1, pp. 22–33, 2009.

Review Article

Analyzing the Classification Techniques for Bulk of Cursive Languages Data: An Overview

Mu Hong ^{1,2}, Shah Nazir ³, Zhang Shuo,^{2,4} and Wang Guan^{2,4}

¹Hohai University School of Public Administration, Nanjing 210000, China

²Sangmyung University the Graduate School, Seoul 03016, Republic of Korea

³Department of Computer Science, University of Swabi, Swabi, Pakistan

⁴Nanhang Jincheng College School of Art and Communication, Nanjing 210016, China

Correspondence should be addressed to Mu Hong; clarke@hhu.edu.cn and Shah Nazir; snsahnzr@gmail.com

Received 28 December 2020; Revised 26 January 2021; Accepted 8 February 2021; Published 23 February 2021

Academic Editor: Muhammad Arif Shah

Copyright © 2021 Mu Hong et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The remarkable growth of texts both in online and offline is becoming a challenging issue which need exploration for further research. Diversities of regional and cultural changes have produced diverse languages as a source of communication. Variations of styles are existing for handwritten texts which is due to varying writing styles. The research area of text recognition is matured which has increased a number of directions in the area of research. A detail report of the existing literature is needed which can help practitioners and researchers to use the existing evidence and provide new solutions for identification of cursive languages and to optimize the ability of recognition for cursive text. For facilitating the researchers and practitioners by providing in-depth analysis of the existing literature, the proposed study provide a detail report through which researchers can get benefit of the literature and devise new solutions. This study is based on searching various popular libraries for identifying relevant materials associated with the proposed study.

1. Introduction

With the passage of time, a significant growth of texts arises both in online and offline. This growth is becoming a challenging issue for researchers which need consideration for further research. Different diversities exist in the form of regional and cultural changes which have produced various languages for communication. The growth of computing devices and facilitations of low-cost access to Internet has presented new directions of retrieving information from digital libraries [1]. In accumulation to this, image, audio, and video archive, the collection of documents be significant part of digital record. For the last couple of years, several organizations have digitized their document collections and have made them available online to facilitate retrieval and for community use. The issue of these documents is that they are mostly in image format which are neither editable and nor searchable. Such documents can occupy more storage space as compared to the textual format. Also, if such images are

accessed/processed through the Internet, it will require more bandwidth. Keeping in view this issue, it was desired by the researchers to convert such images into textual format which can then be easily accessed and processed.

Optical character recognition (OCR) system can facilitate the conversion of document image into textual format. With the advancements, the OCR system for many scripts/languages is in the early stage of research. Such languages include Urdu, Pashto, Persian, Arabic, and many others. With the help of handwriting recognition (HWR), the written text can be transformed into a symbolic depiction. This transformation can facilitate the interaction of human and computer applications like mail sorting, cheque verification, image recognition, office automation, and interaction of human computer [2–4]. The language recognition of handwriting recognition of Latin and Chinese has been researched and has achieved significant success. Parallel to this, the research in other languages like HWR of Urdu, Arabic, Persian, and Pashto is less. The reason is that there

are more variations of writing styles and complexity. The handwriting recognition can be categorized into offline and online systems. The offline text recognition is difficult to recognize due to the reasons that these are available in the form of images with written text, while the online recognition of text is easy; as in such system, there is no need of sequence or order of writing [5].

The field of text recognition is matured which has amplified the directions in the area of research. Detail of the existing literature is desirable which can assist practitioners and researchers to use the existing evidences and provide new solutions for identification of cursive languages and to optimize the ability of recognition for cursive text. For facilitating the researchers and practitioners by providing in-depth analysis of the existing literature, the proposed study provide a detail report through which researchers can get benefit of the literature and devise new solutions.

The organization of the paper is as follows: Section 2 shows the related work to the cursive script and language recognition, and the approaches, techniques, and methods used are described. Section 3 shows the analysis of the existing work associated with the text recognition. The paper is concluded in Section 4.

2. Approaches for Cursive Script Recognition

Researchers are trying to devise algorithms, novel approaches, and solutions for the recognition of cursive script and languages. Shaikh and Shaikh [6] proposed an algorithm of parallel thinning for cursive or noncursive languages by defining a customized set of preservation rules through pixel arrangement grid template, producing strong restriction to noise and speed. The results of experiments showed significant achievements over the other cursive languages such as Sindhi, Urdu, and Arabic and noncursive languages such as Chinese, English, and numerals. Dhande and Kharat [7] presented an approach for cursive language recognition of handwriting in English language. Mostly, in the cursive handwriting of English script, the word characters are connected to each other. So the feature extraction and segmentation of English cursive script is difficult. The approach has used the method of horizontal and vertical projection for segmentation. The algorithm of convex hull is used for extracting features, and support vector machine is used as algorithm for recognition and classification. Chinese language is considered to be a widely used language around the globe. The script of Chinese is the most distinctive traditional culture and calligraphic art of China. Research is needed for its connecting writing recognition for text on cursive images. Qin et al. [8] offered a method for cursive text detection for the dataset and is known as SE-seglink. The feature extraction from image is enhanced through this method. The authors designed a dataset containing 523 images for Chinese cursive text. Comparing to the available approaches, the offered approach is performing better in terms of recognizing the cursive images. The effectiveness of the approach is tested through performing comparative experiments.

Ueki et al. [9] proposed an approach for recognition of consecutive Kuzushiji characters through multiple candidate regions as input to a neural network. An assessment

through database of images of three consecutive Kuzushiji characters confirmed that the approach proposed is having greater rate of accuracy compared to the approach in which the character of images were cropped according to the boundary detected. Han and Sethi [10] proposed a method which uses heuristic rule set for determining probable boundaries of letter in the image with word curved. The heuristic rules are based on the relations existing between assured topologic and geometric features and the character of English language. A system of segmentation has been built which integrates the proposed system for performing segmentation on postal address images. Various steps are involved in the preprocessing of extracting handwritten words from postal envelope and the step of normalization for allowing variation in thickness of pen and witting tilt. The results obtained from the experiments revealed that the approach is efficient and able for locating the boundaries of letter in cursive words accurately. Kim and Lee [11] offered an approach of unified network for recognition of handwritten text in different languages. The system can be used for any grouping of phonetic writing systems such as Japanese, Arabic, and Tai.

Sternby and Friberg [12] presented an approach for interaction of dictionary in recognition of online cursive script. With the help of segmentation graph, all the probable paths are retrieved for corresponding to words in a dictionary in an effective way. The study also deals with the treating of secondary strokes in online segmentation graph. The approach was tested with huge data with good results. Ahmad et al. [13] proposed an approach for finding out the alternate recognizable unit in the cursive script of Pashto. The alternatives are primary ligature and ligature. A corpus of 2313736 words of Pashto is extracted from different sources of web, and 19268 unique ligatures were identified in the cursive script of Pashto. The results showed that 7000 ligatures showed 91% portion of corpus of Pashto words, and 7681 primary ligatures were identified representing the shapes of all the ligatures. Hassan [14] proposed a system for recognition of cursive Arabic writing. The issues arise due to the personal attitude, style variations, and various levels of writing. The system is based on recognition of hierarchal strategy. With the incorporation of grammar and parser, a system of linguistic recognition was developed. Hashemi et al. [15] designed a recognition system for the Persian text. The system contains a stage of segmentation to separate the character that is constituent. This stage is beneficial for italic or highly declined Latin text. The study presented a segmentation algorithm with two steps. In the first step, nonoverlapped and separate isolated characters are separated, while the second step segments nonassociated characters that are overlapped. The approach was tested on the script of real world and showed an accuracy of 99.7%.

3. Analyzing the Existing Studies for Cursive Script/Languages

The following subsections present the related work and analysis of the cursive script/languages.

3.1. Existing Research for Supporting Recognition of Cursive Script. Recognition of cursive languages in effective way is become challenging issue for researchers and practitioners. Diverse approaches have been proposed to tackle the issue of cursive languages from different perspectives. Erdogan and Ozge [16] suggested a study for analyzing the cursive handwriting of likely primary school teachers from the legibility viewpoint. The nature of the study is perspective with the aim to portray the available state of affairs, making use of qualitative methods. The study consist of 130 potential primary school teachers and were asked for copying the text presented by researchers using cursive handwriting. The cursive handwriting was examined through “cursive handwriting basement form.” The study showed that the handwriting of teachers was sufficiently legible. Samanta et al. [17] proposed a hidden Markov model-based online unconstrained word recognition of handwritten samples. The system involved the key steps which are the handwriting segmentation into substrokes, extracting features from substrokes and recognition. For the task of segmentation, a strategy of discrete curve evolution is proposed. Then various linear and angular features are extracted from substrokes of samples of word and are modelled as feature vectors produced from a mixture distribution. The algorithm of Baum–Welch parameter estimation was used for handling the spherical linear correlated data for constructing the hidden Markov model. At last, the recognition classifier was designed for handwritten word samples. The results demonstrated that Bangla and Latin scripts have good performance of the suggested scheme of recognition. Camastra [18] proposed a recognizer approach for cursive characters which is a module in the recognition of any cursive word based on the approach of segmentation and recognition. With the help of support vector machine and neural gas, the classification of character is achieved. For verification of lower and upper case version of various letters, the neural gas was used, while for recognition of character, the SVM was used. The dataset of 57293 characters was considered for training and testing of the recognizer for cursive characters. The results reveal good performance by the use of SVM and showed better efficiency.

Darwish and ELgohary [19] presented an approach of bio-inspired expert system for printer forensics that incorporate both the features of texture and niching genetic search for selecting effective sufficient minimized feature set. The approach k-nearest neighbours was used for differentiating the printer brand for its simplicity. The results reveal that the approach is having high accuracy of classification and can take less time. Wen et al. [20] proposed a model of hierarchal deformation for describing the online cursive Chinese character deformation. The approach includes two levels: firstly, matching two sequences of turn points which are extracted from the reference and input characters for describing the matches of stroke. Then, the constrained parabola transformation is used for reducing the difference between the matched strokes correctly. The results show that hierarchal deformation approach is effective to the deformation of cursive Chinese character with less computational cost. Lee and Verma [21] proposed a new binary

segmentation algorithm for reducing the issue of chain failure risk in the course of validation and improved the accuracy of segmentation. The binary segmentation algorithm is a combination segmentation approach including validation and over-segmentation. The validity of the research was carried out on benchmark of CEDAR database, and the results showed better performance. EL-SHEIKH and GUINDI [22] designed a recognition system for Arabic text. The approach contains segmentation stage for recognition of Arabic cursive words that are typewritten. The system showed 99% recognition rate. Bhunia et al. [23] presented an approach of cross language platform for recognition and spotting of handwritten word. The approach is presented for scripts of low resource where training is done with huge dataset of an accessible script and test is done on the other script. The approach was tested on three Indic scripts including Devanagari, Bangla, and Gurumukhi.

Chandio et al. [24] presented a dataset for the detection of Urdu text, and recognition in natural scene images is analyzed. Above 2500 natural scene images were collected for developing dataset through the digital camera and with a mobile camera. Three datasets were developed including cropped word images, isolated Urdu character images, and end-to-end text spotting. The emphasis was given to the Urdu text instances. The approach can be used for performing detection and recognition of Urdu text as well as end-to-end recognition in natural scenes. Development of these datasets can provide help in developing Persian and Arabic natural scene text recognition and detection. Aisyah et al. [25] carried out a research for designing and developing learning materials for students for learning Japanese language as foreign language at the Universiti Kebangsaan Malaysia. The research carried out a survey of commercially produced text book and preproduction stage focusing on needs analysis from students. Abuhaiba [26] presented an approach for identification of cursive language or discrete script enclosed in an image document. The approach is based on extraction of set of global templates shared between languages and scripts with common shapes of symbols. It saves time of processing and requirement of memory in the execution of program. The approach performed one-dimensional normalization like retaining width-to-height ratio. The authors recommended the approaches that have good accuracy and speed for commercial use of OCR products Table 1 shows some of the approaches/methods used for recognition of cursive script/languages from different perspectives.

3.2. Analyzing the Literature for Cursive Script Recognition. Recognition of cursive script/language is considered to be important for different purposes. This recognition of cursive script from the images can save time and storage memory. As these are in textual format, different approaches have been proposed. AlKhateeb et al. [37] have used the hidden Markov models for recognition of word-based offline text. Three stages are involved in the method including pre-processing, feature extraction, and classification. Initially, the words from the scripts of input are segmented and normalized, then features are extracted from the segmented

TABLE 1: Approaches used for cursive text/script/language recognition.

Reference	Authors	Method	Year	Article type
[27]	J. B. Hellige and M. M. Adamson	Hemispheric differences in processing handwritten cursive	2007	Journal
[28]	A. Jalali and M. Lee	High cursive traditional Asian character recognition using integrated adaptive constraints in ensemble of DenseNet and inception models	2020	Journal
[29]	W. Cho, S.-W. Lee, and J. H. Kim	Modeling and recognition of cursive words with hidden Markov models	1995	Journal
[30]	S. Naz et al.	Offline cursive Urdu-Nastaliq script recognition using multidimensional recurrent neural networks	2016	Journal
[31]	M. Schambach	Recurrent HMMs and cursive handwriting recognition graphs	2009	Conference
[32]	T. G. Rose and L. J. Evett	Semantic analysis for large vocabulary cursive script recognition	1993	Conference
[33]	R. J. Kannan, R. Prabhakar, and R. M. Suresh	Off-line cursive handwritten Tamil character recognition	2008	Conference
[34]	A. A. Chandio, M. Asikuzzaman, and M. R. Pickering	Cursive character recognition in natural scene images using a multilevel convolutional neural network fusion	2020	Journal
[35]	J. Danna, D. Massendari, B. Furnari, and S. Ducrot	The optimal viewing position effect in printed versus cursive words: Evidence of a reading cost for the cursive font	2018	Journal
[36]	B. Verma and H. Lee	Segment confidence-based binary segmentation (SCBS) for cursive handwritten words	2011	Journal

words, and then these features are integrated for classification purpose. The database of IFN/ENIT was used which contains 32492 words of Arabic handwritten notes. The approach delivered better performance compared to the existing approaches. Abu-Ain et al. [38] proposed a detection approach for baseline and straightness for the text of cursive handwritten notes. The approach is based on the analysis and extraction of directions features from subwords of the text skeleton. The text of Arabic language was considered as a case study. The results revealed that the approach is efficiently working and tested on Arabic dataset. Mouhcinea et al. [39] proposed a method of Arabic handwritten cursive text based on the hidden Markov model. The experimental results of the images of IFN/ENIT database benchmark revealed that the suggested approach enhanced recognition. Manjusha et al. [40] proposed an approach which aimed for building the databases of handwritten character image for the script of Malayalam language. The samples of handwritten collected from 77 native Malayalam writers. The contour model-based image segmentation algorithm was used for extracting the character images from the data sheets of handwritten. Features extraction techniques were used for extracting features. The scattering convolution network-based feature descriptors achieved a recognition accuracy of 91.05% which is the highest among the available feature descriptors. Naz et al. [41] reviewed the literature on OCR associated with the Urdu cursive scripts. The Pashto, Urdu, and Sindhi languages are described with focus on the script of Nasta'liq and Naskh.

Apart from the above literature, various popular libraries such as ScienceDirect, IEEE, Springer, and Wiley Online

were used to search for achieving the most relevant materials. These libraries were only considered due to the reasons that these are only publishing peer reviewed and quality research. Figure 1 shows the initial results of the search process in the mentioned libraries. The figure shows that more materials were obtained in the library of Springer followed by the ScienceDirect.

Initially the library of ScienceDirect was searched, and the results were depicted in figures. Figure 2 shows the publication titles with the number of articles.

Figure 3 represents the type of articles in the given library.

Figure 4 represents the number of articles in the given years.

Figure 5 shows the subject areas in the given libraries.

The library of IEEE was searched and the topics of articles are shown in Figure 6.

The article type is shown in Figure 7 where most of the papers are published as conference papers.

The locations where the conferences held were identified in the study conducted. Figure 8 depicts the locations of conferences held.

After this, the library of Springer was searched for identification of relevant materials and their analysis. Figure 9 represents the various disciplines of articles. The figure shows that more articles were published in the area of Computer Science.

Figure 10 shows the subdisciplines of the areas with total of publications.

The article type was identified in the given library. Figure 11 shows the total number of publications based on the article types.

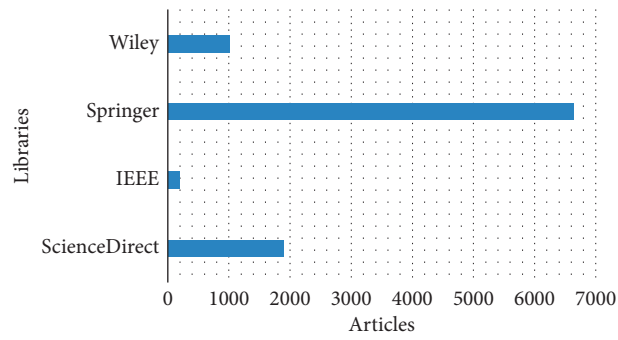


FIGURE 1: Search process of libraries.

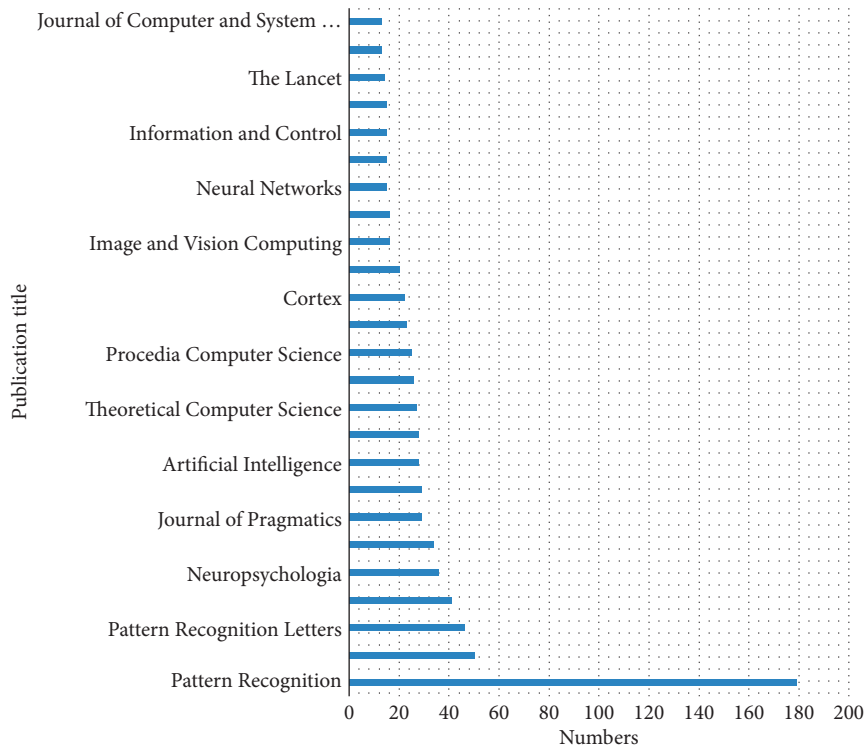


FIGURE 2: Publication title along with the papers.

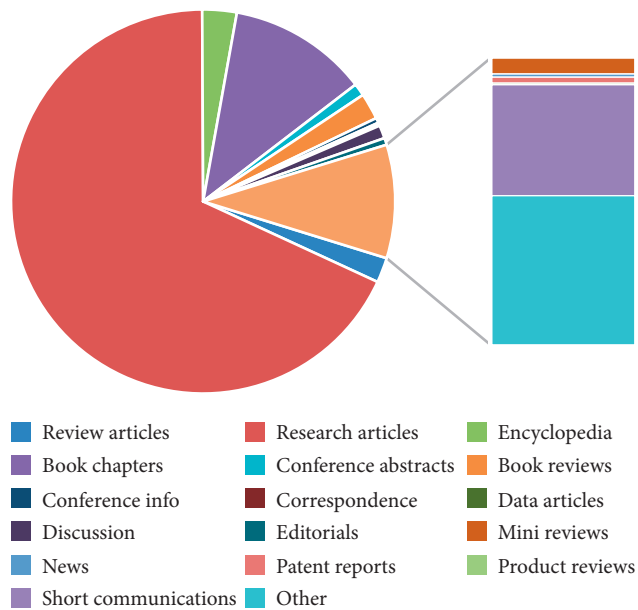


FIGURE 3: Type of articles.

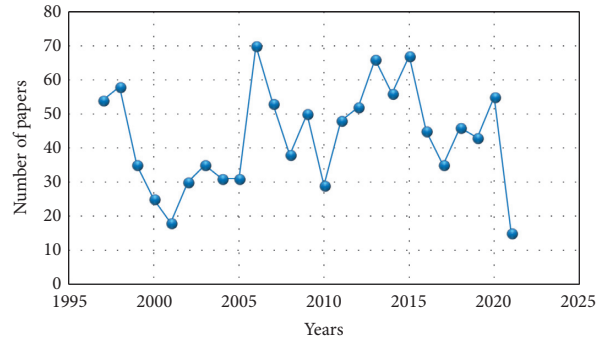


FIGURE 4: Number of publications in the given years.

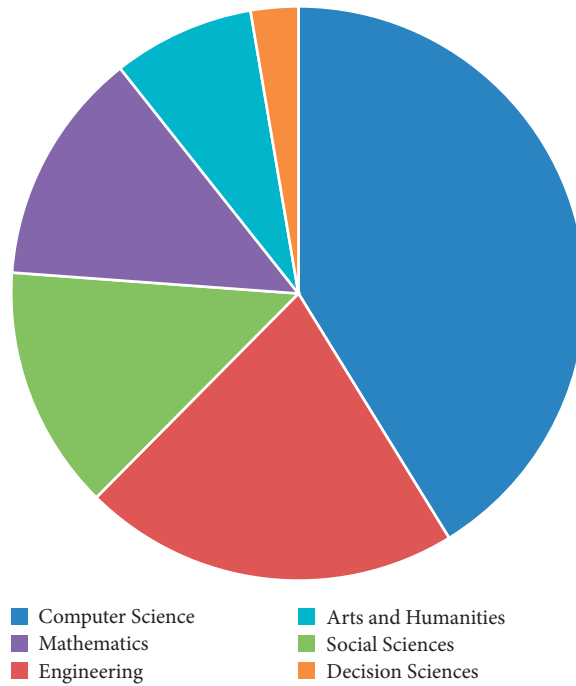


FIGURE 5: Subject areas.

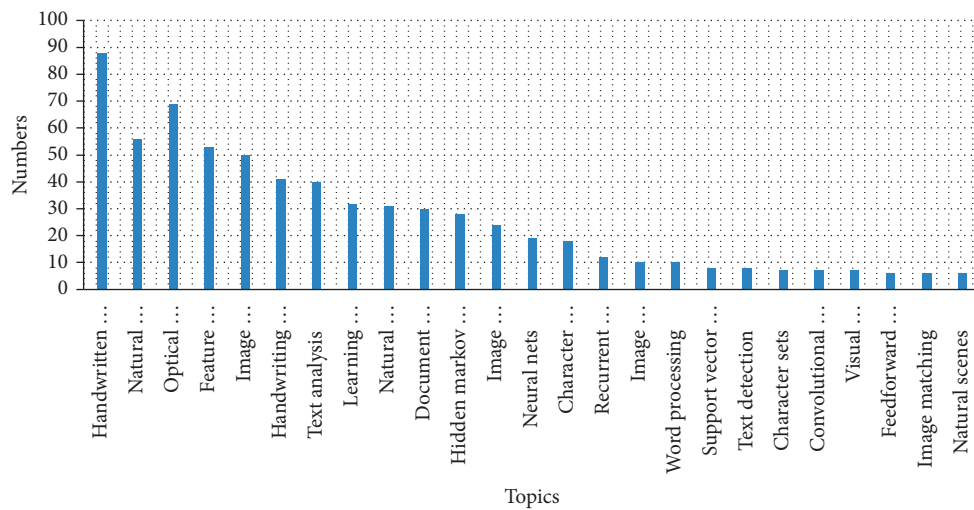


FIGURE 6: Topics of publications.

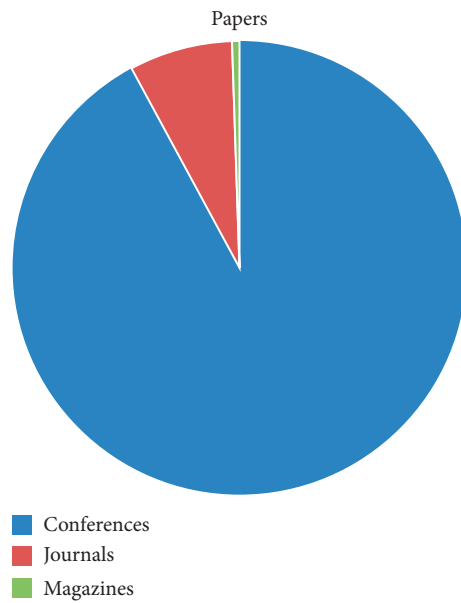


FIGURE 7: Publication types.

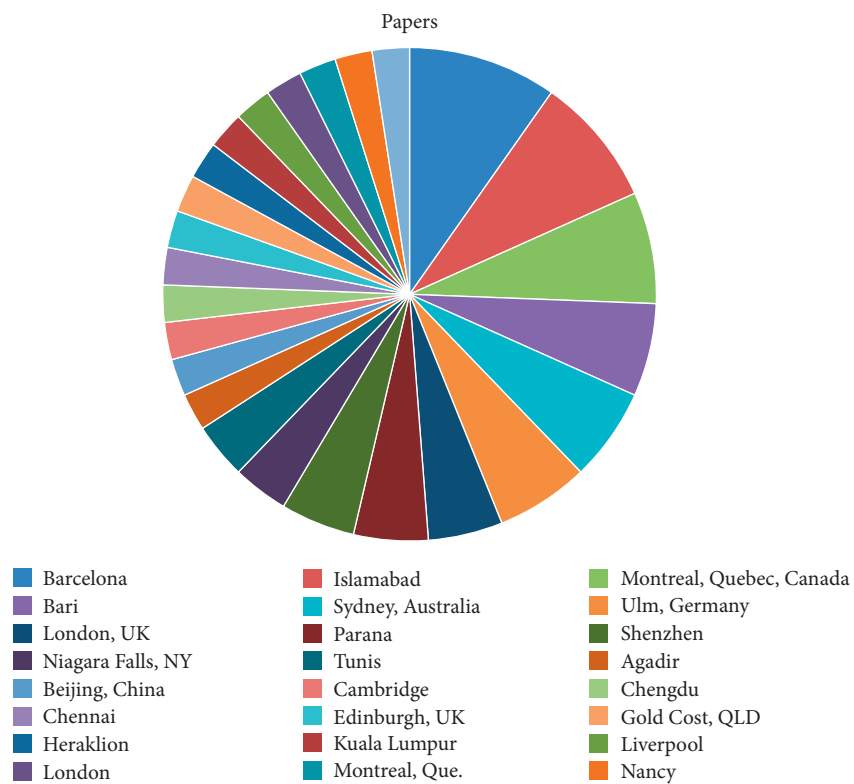


FIGURE 8: Location of conferences held.

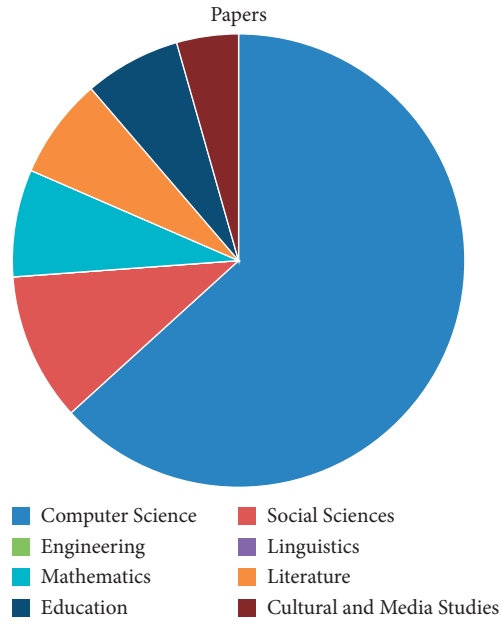


FIGURE 9: Discipline of publications.

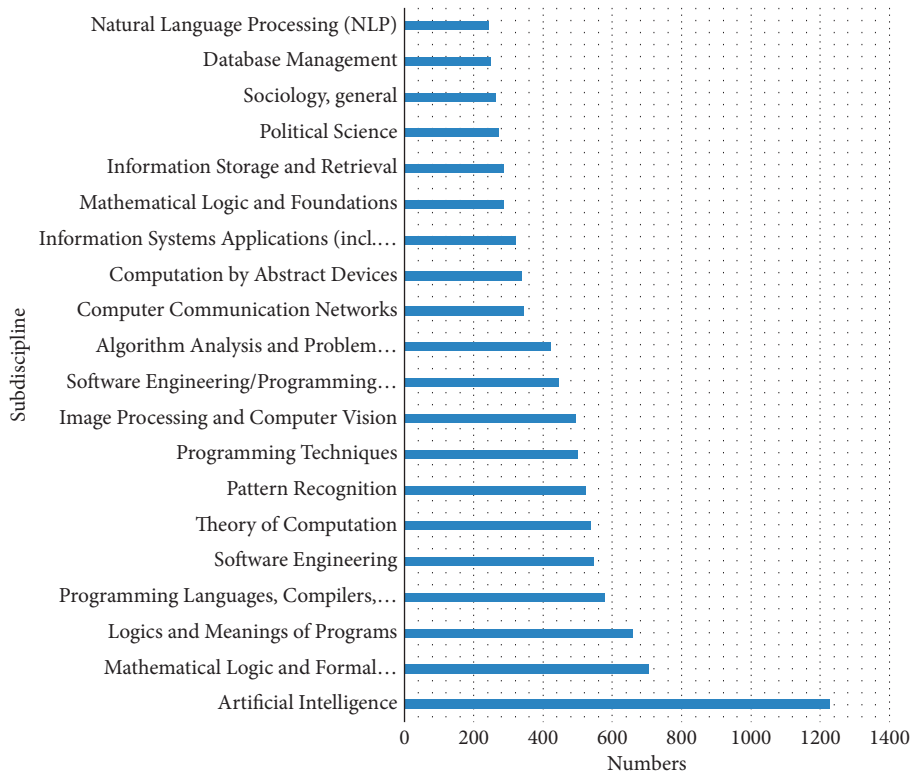


FIGURE 10: Subdisciplines of the areas with publications.

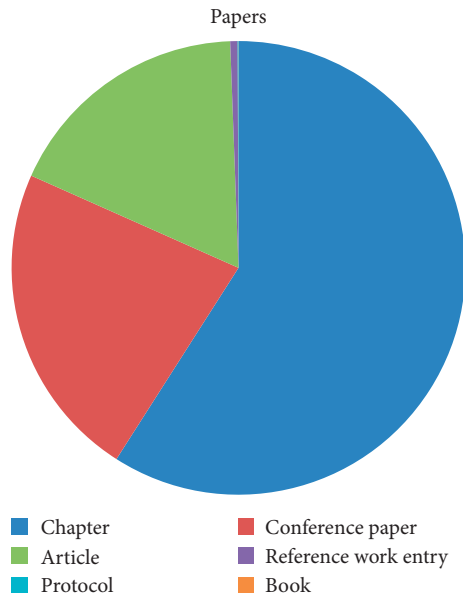


FIGURE 11: Article types with publications.

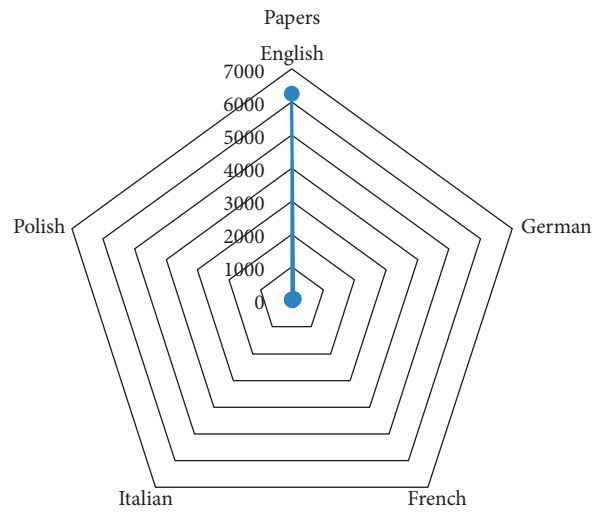


FIGURE 12: Languages of publications.

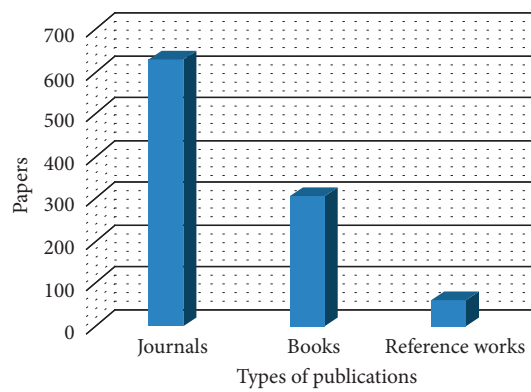


FIGURE 13: Types of publications with total numbers.

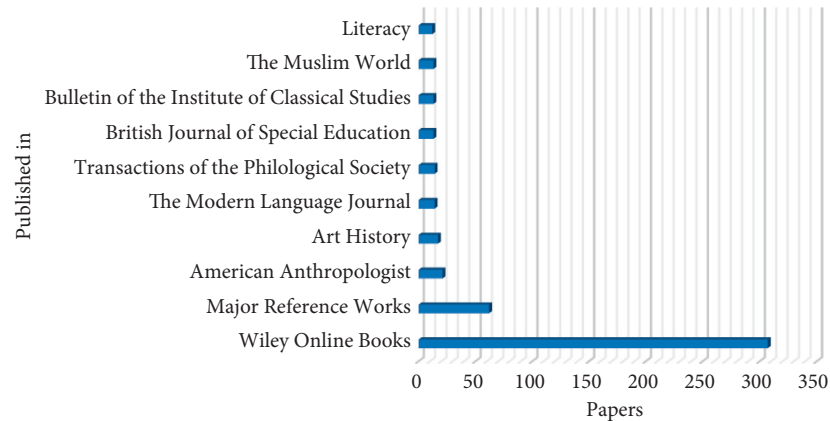


FIGURE 14: Papers published in with the number of articles.

The study also focused on the language of publications. Figure 12 shows the language of articles with the total number.

Figure 13 depicts the type of publications in the Wiley online library.

Figure 14 represents the papers published in with the number of articles.

4. Conclusion

With the recent advancements of modern technology and innovation in the field of machine translation, a significant growth of texts arises both in online and offline scripts. This growth is becoming a challenging issue for researchers which need consideration for further research and exploration. Diversities exist in the form of regional and cultural changes. The diversities of regional and cultural changes have produced diverse languages as a source of communication. Variations of styles exist for handwritten texts which are due to varying writing styles. The research area of text recognition is matured which has increased the directions in the area of research for exploration. A detail report of the existing literature is necessary which can support practitioners and researchers to use the existing literature as evidence and provide new solutions for identification of cursive languages and to optimize the ability of recognition for cursive text. The current study has provided a detail report through which researchers can get benefit of the literature and devise new solutions. The study is further facilitating the researchers and practitioners by providing in-depth analysis of the existing literature.

Data Availability

No data are available.

Conflicts of Interest

The authors declare no conflicts of interest.

References

- [1] A. Bhardwaj, A. Thomas, Y. Fu, and V. Govindaraju, "Retrieving hand writing styles: a content based approach to hand written document retrieval," in *Proceedings of the 2nd International Conference on Frontiers in Handwriting Recognition (ICFHR10)*, pp. 265–270, Kolkata, India, November 2010.
- [2] S. Alma'Adeed, C. Higgins, and D. Elliman, "Off-line recognition of handwritten Arabic words using multiple hidden Markov models," *Knowledge Based Systems*, vol. 17, pp. 75–79, 2004.
- [3] Y. Kessentini, T. Paquet, and A. M. Benhamadou, "Multi-script handwriting recognition with n-streams low level features," in *Proceedings of the 19th Internet. Conf. Pattern Recognition (ICPR)*, Tampa, FL, USA, December 2008.
- [4] M. R. Al-Hajj, L. Likforman-Sulem, and C. Mokbel, "Combining slanted-frame classifiers for improved HMM-based Arabic handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, pp. 1165–1177, 2009.
- [5] A. Amin, "Off-line Arabic character recognition," *Pattern Recognition*, vol. 31, no. 5, pp. 517–530, 1998.
- [6] N. A. Shaikh and Z. A. Shaikh, "A generalized thinning algorithm for cursive and non-cursive language scripts," in *Proceedings of the 2005 Pakistan Section Multitopic Conference*, pp. 1–4, Karachi, Pakistan, December 2005.
- [7] P. Dhande and R. Kharat, "Recognition of cursive english handwritten characters," in *Proceedings of the 2017 International Conference on Trends in Electronics and Informatics (ICEI)*, pp. 199–203, Tirunelveli, India, May 2017.
- [8] X. Qin, J. Jiang, W. Fan, and C. Yuan, "Chinese cursive character detection method," *The Journal of Engineering*, vol. 2020, no. 13, pp. 626–629, 2020.
- [9] K. Ueki, T. Kojima, R. Mutou, R. S. Nezhad, and Y. Hagiwara, "Recognition of Japanese connected cursive characters using multiple softmax outputs," in *Proceedings of the 2020 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pp. 127–130, Guangdong, China, August 2020.
- [10] H. Ke and I. K. Sethi, "Off-line cursive handwriting segmentation," in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, pp. 894–897, Montral, Canada, August 1995.

- [11] J. H. Kim and J. J. Lee, "A unified network-based approach for recognition of cursive handwritings in mixed languages: a case study on Hangul and Roman mixture," in *Proceedings of the IEEE Colloquium on Handwriting and Pen-Based Input*, pp. 6/1–6/4, London, UK, March 1994.
- [12] J. Sternby and C. Friberg, "The recognition graph-language independent adaptable on-line cursive script recognition," in *Proceedings of the Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, pp. 14–18, Seoul, South Korea, August 2005.
- [13] R. Ahmad, M. Z. Afzal, S. F. Rashid, M. Liwicki, A. Dengel, and T. Breuel, "Recognizable units in Pashto language for OCR," in *Proceedings of the 2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1246–1250, Tunis, Tunisia, August 2015.
- [14] N. Hassan, "Recognition of Arabic cursive handwriting," in *Proceedings of the Geometric Modeling and Imaging--New Trends (GMAI'06)*, pp. 135–140, London, UK, July 2006.
- [15] M. R. Hashemi, O. Fatemi, and R. Safavi, "Persian cursive script recognition," in *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, pp. 869–873, Montreal, Canada, August 1995.
- [16] T. Erdogan and O. Erdogan, "An analysis of the legibility of cursive handwriting of prospective primary school teachers," *Procedia - Social and Behavioral Sciences*, vol. 46, pp. 5214–5218, 2012.
- [17] O. Samanta, A. Roy, S. K. Parui, and U. Bhattacharya, "An HMM framework based on spherical-linear features for online cursive handwriting recognition," *Information Sciences*, vol. 441, pp. 133–151, 2018.
- [18] F. Camastra, "A SVM-based cursive character recognizer," *Pattern Recognition*, vol. 40, no. 12, pp. 3721–3727, 2007.
- [19] S. M. Darwish and H. M. ElGohary, *Building an Expert System for Printer Forensics: A New Printer Identification Model Based on Niching Genetic Algorithm*, Wiley, Hoboken, NJ, USA, 2020.
- [20] W.-T. Chen and T.-R. Chou, "A hierarchical deformation model for on-line cursive script recognition," *Pattern Recognition*, vol. 27, no. 2, pp. 205–219, 1994.
- [21] H. Lee and B. Verma, "Binary segmentation algorithm for english cursive handwriting recognition," *Pattern Recognition*, vol. 45, no. 4, pp. 1306–1317, 2012.
- [22] T. S. El-Sheikh and R. M. Guindi, "Computer recognition of Arabic cursive scripts," *Pattern Recognition*, vol. 21, no. 4, pp. 293–302, 1988.
- [23] A. K. Bhunia, P. P. Roy, A. Mohta, and U. Pal, "Cross-language framework for word recognition and spotting of Indic scripts," *Pattern Recognition*, vol. 79, pp. 12–31, 2018.
- [24] A. A. Chandio, M. Asikuzzaman, M. Pickering, and M. Leghari, "Cursive-text: a comprehensive dataset for end-to-end Urdu text recognition in natural scene images," *Data in Brief*, vol. 31, Article ID 105749, 2020.
- [25] A. Aisyah, N. Hieda, M. Nezu, and N. Ibrahim, "Designing hiragana learning materials for Japanese language course in UKM," *Procedia-Social and Behavioral Sciences*, vol. 59, pp. 451–458, 2012.
- [26] I. S. I. Abuhaiba, "Discrete script or cursive language identification from document images," *Journal of King Saud University - Engineering Sciences*, vol. 16, no. 2, pp. 253–268, 2004.
- [27] J. Hellige and M. Adamson, "Hemispheric differences in processing handwritten cursive☆," *Brain and Language*, vol. 102, no. 3, pp. 215–227, 2007.
- [28] A. Jalali and M. Lee, "High cursive traditional Asian character recognition using integrated adaptive constraints in ensemble of densenet and Inception models," *Pattern Recognition Letters*, vol. 131, pp. 172–177, 2020.
- [29] W. Cho, S.-W. Lee, and J. H. Kim, "Modeling and recognition of cursive words with hidden Markov models," *Pattern Recognition*, vol. 28, no. 12, pp. 1941–1953, 1995.
- [30] S. Naz, A. I. Umar, R. Ahmad et al., "Offline cursive Urdu-Nastaliq script recognition using multidimensional recurrent neural networks," *Neurocomputing*, vol. 177, pp. 228–241, 2016.
- [31] M. Schambach, "Recurrent HMMs and cursive handwriting recognition graphs," in *Proceedings of the 2009 10th International Conference on Document Analysis and Recognition*, pp. 1146–1150, Barcelona, Spain, July 2009.
- [32] T. G. Rose and L. J. Evett, "Semantic analysis for large vocabulary cursive script recognition," in *Proceedings of 2nd International Conference on Document Analysis and Recognition (ICDAR)*, pp. 236–239, Tsukuba, Japan, October 1993.
- [33] R. J. Kannan, R. Prabhakar, and R. M. Suresh, "Off-line cursive handwritten Tamil character recognition," in *Proceedings of the 2008 International Conference on Security Technology*, pp. 159–164, Hainan Island, China, December 2008.
- [34] A. A. Chandio, M. Asikuzzaman, and M. R. Pickering, "Cursive character recognition in natural scene images using a multilevel convolutional neural network fusion," *IEEE Access*, vol. 8, pp. 109054–109070, 2020.
- [35] J. Danna, D. Massendari, B. Furnari, and S. Ducrot, "The optimal viewing position effect in printed versus cursive words: evidence of a reading cost for the cursive font," *Acta Psychologica*, vol. 188, pp. 110–121, 2018.
- [36] B. Verma and H. Lee, "Segment confidence-based binary segmentation (SCBS) for cursive handwritten words," *Expert Systems with Applications*, vol. 38, no. 9, pp. 11167–11175, 2011.
- [37] J. H. AlKhateeb, J. Ren, J. Jiang, and H. Al-Muhtaseb, "Offline handwritten Arabic cursive text recognition using Hidden Markov Models and re-ranking," *Pattern Recognition Letters*, vol. 32, no. 8, pp. 1081–1088, 2011.
- [38] T. Abu-Ain, S. N. H. S. Abdullah, B. Bataineh, W. Abu-Ain, and K. Omar, "Text normalization framework for handwritten cursive languages by detection and straightness the writing baseline," *Procedia Technology*, vol. 11, pp. 666–671, 2013.
- [39] R. Mouhcine, A. Mustapha, and M. Zouhir, "Recognition of cursive Arabic handwritten text using embedded training based on HMMs," *Journal of Electrical Systems and Information Technology*, vol. 5, no. 2, pp. 245–251, 2018.
- [40] K. Manjusha, M. A. Kumar, and K. P. Soman, "On developing handwritten character image database for Malayalam language script," *Engineering Science and Technology, an International Journal*, vol. 22, no. 2, pp. 637–645, 2019.
- [41] S. Naz, K. Hayat, M. Imran Razzak, M. Waqas Anwar, S. A. Madani, and S. U. Khan, "The optical character recognition of Urdu-like cursive scripts," *Pattern Recognition*, vol. 47, no. 3, pp. 1229–1248, 2014.

Research Article

Software Birthmark Usability for Source Code Transformation Using Machine Learning Algorithms

Keqing Guan,¹ Shah Nazir ,² Xianli Kong ,³ and Sadaqat ur Rehman⁴

¹Institute for Big Data Research, Liaoning University of International Business and Economics, Dalian 116052, China

²Department of Computer Science, University of Swabi, Swabi, Pakistan

³School of Economics, Dongbei University of Finance & Economics, Dalian 116025, China

⁴Department of Computer Science, Namal Institute, Mianwali 42250, Pakistan

Correspondence should be addressed to Shah Nazir; snshahnzr@gmail.com and Xianli Kong; kongxianli@dufe.edu.cn

Received 14 January 2021; Revised 24 January 2021; Accepted 30 January 2021; Published 9 February 2021

Academic Editor: Sikandar Ali

Copyright © 2021 Keqing Guan et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Source code transformation is a way in which source code of a program is transformed by observing any operation for generating another or nearly the same program. This is mostly performed in situations of piracy where the pirates want the ownership of the software program. Various approaches are being practiced for source code transformation and code obfuscation. Researchers tried to overcome the issue of modifying the source code and prevent it from the people who want to change the source code. Among the existing approaches, software birthmark was one of the approaches developed with the aim to detect software piracy that exists in the software. Various features are extracted from software which are collectively termed as “software birthmark.” Based on these extracted features, the piracy that exists in the software can be detected. Birthmarks are considered to insist on the source code and executable of certain programming languages. The usability of software birthmark can protect software by any modification or changes and ultimately preserve the ownership of software. The proposed study has used machine learning algorithms for classification of the usability of existing software birthmarks in terms of source code transformation. The K-nearest neighbors (K-NN) algorithm was used for classification of the software birthmarks. For cross-validation, the algorithms of decision rules, decomposition tree, and LTF-C were used. The experimental results show the effectiveness of the proposed research.

1. Introduction

Source code transformation is performed in a manner in which the source code of a program is transmuted by spotting any operation for creating an alternative or nearly same program. This is mostly performed in situation of piracy where the pirates want the ownership of the software program. From different perspectives, the transformed source code is mostly equivalent to the original program in terms of semantics. For transforming the source into another program, one usually needs the incorporation of whole front end of programming language, data structure of internal program representation, parsing of source code, understanding of the program, meaningful static analysis, and generation of useable source code for representation of

program. Software industry is immensely in front of the software piracy issues. This piracy performed in software can badly affect the software business and eventually big loss to the owner organizations. Stoppage of software piracy is extremely important for the rising economy of the software industry. Different methods are used to prevent piracy of software. These methods include techniques of fingerprinting [1, 2], watermarking [3–5], and software birthmarks [6–11]. The watermark has the weaknesses as it can be removed by approaches of code obfuscations and semantic preserving transformation. The similar concerns are existing in the software fingerprints. To overawe these limitations, the idea of birthmark was presented and is broadly acknowledged and known approach for preventing source code transformation and piracy of software.

Birthmark of software is considered as necessary features which can be employed for focusing the identification and uniqueness of software. The common uses of birthmarks are for software theft, identification of transformations in source code, and Windows API. More features of a software birthmark can eventually present the robustness and effectiveness which will further show the precise detection of transformations or theft made in the software or program. Birthmark of software is established on imperative properties, resilience, and credibility [6]. Credibility depicts that the birthmark of software entails that two programs, which is written independently, should be different. Whereas, the resilience should be preserved and not be damaged in any case. Various approaches were considered to show the effectiveness and usability of software birthmark [6, 12–14]. These approaches talk about various applications of software birthmark including source code transformation, code obfuscations, software theft, piracy, and many others. The use of software birthmark can protect software by any adaptation and ultimately preserve the ownership of software.

The proposed study endeavored to use machine learning algorithms for classification of the usability of existing software birthmarks in terms of source code transformation. The K-nearest neighbors algorithm was used for classification of the software birthmarks. For cross-validation, the algorithms of decision rules, decomposition tree, and LTF-C were used. The experimental results show the effectiveness of the proposed research.

This study is divided into different sections. Section 2 represents the related work associated to the existing approaches of source code transformation, software theft, and so on. The research methodology of the proposed study is presented in Section 3. Results and Discussion are given in Section 4. The study is concluded in Section 5.

2. Related Work

Researchers frequently attempt to devise diverse approaches, methods, and solutions to proficiently and successfully analyze the source code transformation and software piracy. Numerous practices have been adopted in software industry to detect and prevent software theft. The idea of software birthmark was presented to overcome the downsides of software fingerprint, watermarks, and digital signature as these can be modified or removed by using approaches of code obfuscation and transformation of semantic preservation. Birthmark of software was established to powerfully recognize the software theft. First, the birthmark was developed by Tamada et al. [15], which extracts four types of birthmarks: inheritance structure, sequence of method calls, constant values in field variables, and the used classes. With the advancements in the field, birthmark was well thought out as a significant measure of the software in serving to identify software piracy. The field was discovered, and lots of researchers tried to grow a strong and further trustworthy birthmark for finding of software piracy. Software birthmark knowledge was initially considered as sole identification of object by Neufeld [16] in 1992. Derrick [17] discovered the idea and gave the importance to the use of birthmark details

for “protecting” software. This was later termed as theft protection of software.

The primary software birthmark was allied with software theft which was offered as a birthmark for Java program theft detection [15]. In the same way in 2004, Tamada et al. considered birthmark of software that was used for detecting the theft in Windows applications. Myles and Collberg proposed “whole program path birthmarks” for detecting software theft [18]. That birthmark method was created on the whole control flow of the software program. Diverse categories of birthmark were planned for software theft detection. The proposed study identified a number of important birthmarks which have been proposed by different researchers for different purposes mostly for theft detection. Spafford and Weeber [19] discovered dissecting executable code for analyzing the structure of data, library calls, and system calls. The idea of software forensics was offered for thoughtful source of virus and malware infection. Birthmark was aimed to facilitate detection of transformation in source code and software theft [20–23]. These studies have considered the design of own birthmark according to some defined features of software and then evaluated the effectiveness in term of creditability and resilience of software for identification of theft exists in copies of software.

The authors [24] offered a dynamic program slicing tool built on dynamic birthmark with some inputs; a union of k-gram instruction-sequence sets as birthmark is used for identification of program. Formal description of software birthmarks was offered by Tamada et al. [25] where they proposed an approach of extracting birthmark from the class files of Java. They are sightseen on comparable perception and proposed a framework for evaluating the two significant properties of birthmarks that is resilience and credibility. Zeng et al. [26] devised a framework of semantic-based abstract interpretation for evaluating software birthmark. This model defines two important properties resilience and credibility. The success of the framework is confirmed by static API birthmark and static-gram birthmark. The authors presented a dynamic birthmark for Java that perceives how a program uses objects providing by the Java standard API [27].

For transmuting the source code into an alternative program, commonly, it needs the integration of data structure of internal program representation, whole front end of programming language, meaningful static analysis, parsing of source code, understanding of the program, and generation of useable source code for representation of the program. Various approaches are being used to change the source code. To overcome this issue, the researchers have devised different solutions. The proposed study has used machine learning algorithms for classification of the software birthmarks usability in terms of source code transformation. The K-nearest neighbors algorithm was used for classification of the software birthmarks.

3. Research Methodology

Efforts are made to overcome the issues raised from the transformation of source code and software theft.

TABLE 1: Software birthmark.

S. no.	Ref.	Approaches of software birthmark
1	[8]	Birthmark-based approach for intellectual software asset management
2	[9]	DKISB
3	[10]	Instruction-words based software birthmark
4	[11]	JSBiRTH
5	[29]	API call structure
6	[15]	CVFV, SMC, IS, and UC
7	[18]	Detecting software theft via whole program path birthmarks
8	[30]	Birthmark-based android application filtering
9	[31]	System call dependence graph
10	[24]	Dynamic k-gram-based software birthmark
11	[32]	k-gram-based software birthmarks
12	[33]	Dynamic key instruction sequences
13	[23]	Features-based software birthmark
14	[34]	Method-based static software birthmarks
15	[35]	CHI-based instruction-words based software birthmark
16	[36]	Thread-aware software birthmarks
17	[37]	System call-based birthmarks

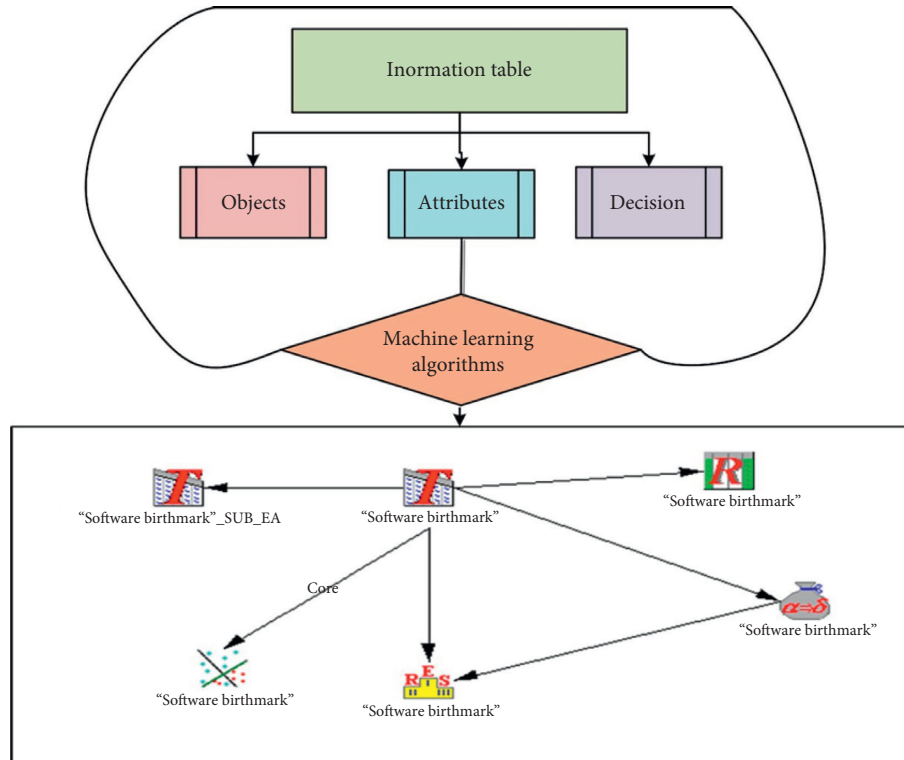


FIGURE 1: Flowchart of the proposed approach.

Researchers mostly considered software ownership and safety as one of the most priorities under consideration. A lot of research studies have been shown for shaping the idea of software birthmarks. Maximum of the birthmark approaches are related to Java source code, which are used for detecting Java theft. Further significant birthmark approaches and techniques works for Windows API [28], for detecting software theft. One of the significant notions used in describing a software birthmark is the usage of software features. Software can be divided into various parts (mostly features) of software [23]. Together, all these features of the

software can deliver a faster and reliable identification of the software and then eventually be used for detection of theft. To detect transformation in source code or software theft, the birthmarks of software applications are matched, and similar birthmark identifies software piracy. A number of birthmarks were identified in the literature. The details are given in Table 1.

Figure 1 represents the flowchart of the approach used in the proposed study for software birthmark usability for source code transformation. The figure represents the information table (dataset) containing objects, attributes,

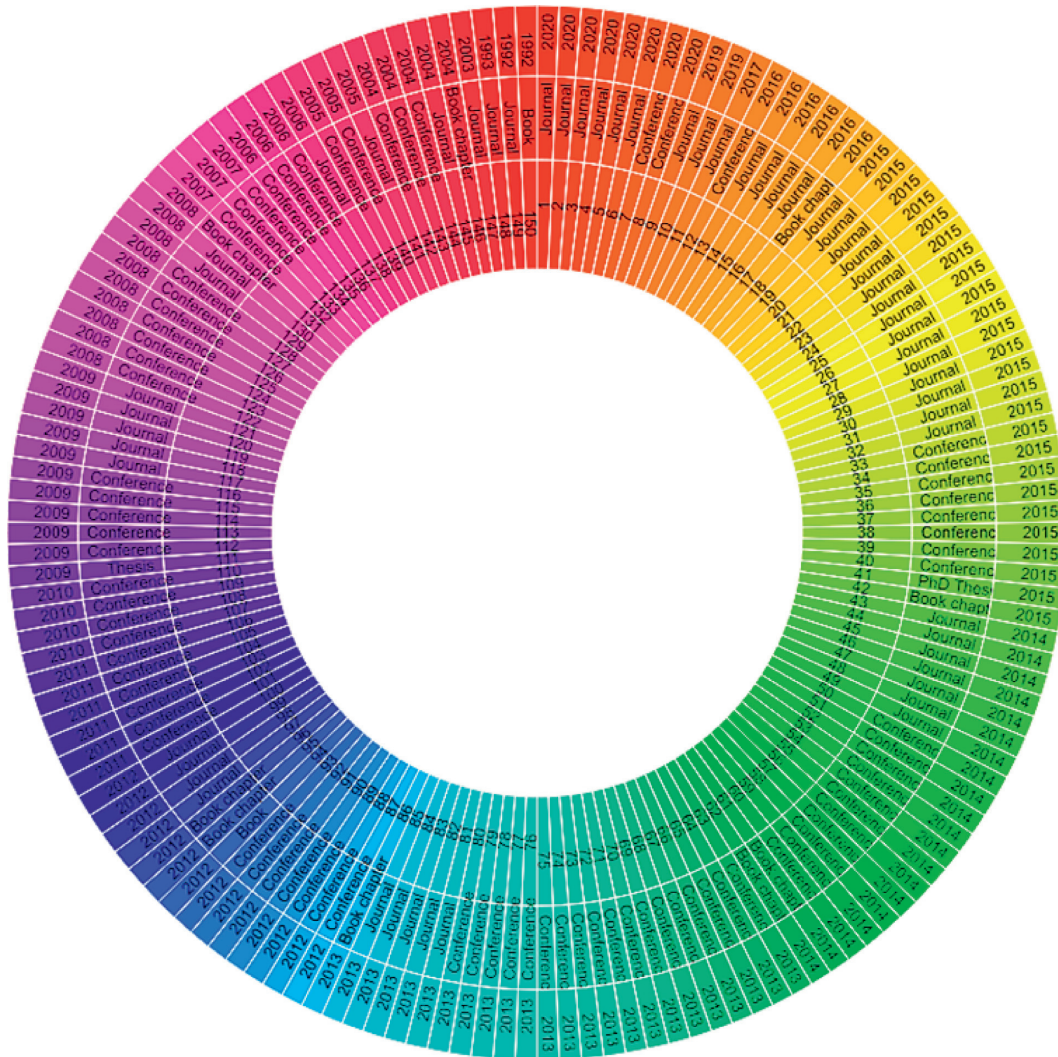


FIGURE 2: Dataset visualization.

and their decision. After the information table, the machine learning algorithms were applied. The K-NN algorithm was applied for the classification purpose. After that, the algorithms of decision rules, decomposition tree, and LTF-C algorithms were applied as cross-validation algorithms.

The dataset developed during higher studies programme was considered for validation purpose of the proposed research. Total of 150 entries were existing with three features. Figure 2 shows the visualization of the dataset for user understanding.

Once the information table was imported to the proposed system, initially, the reduct was applied. After that, rules set were generated. Figure 3 depicts the publications with the year in the given dataset.

Figure 4 depicts the rules set generated from the proposed study.

After doing this process, in last, the algorithm of K-NN was applied to the proposed research. Some cross-validation algorithms were used which are discussed in the Results and Discussion Section.

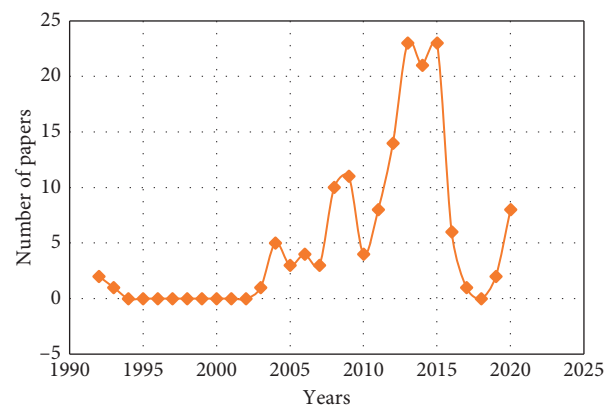


FIGURE 3: Publications with years based on dataset.

4. Results and Discussion

Several research studies have been conducted for refining software birthmarks for detection of piracy. The existing approaches used for detection of piracy are given as

(1-109)	Match	Decision rules
1	8	(Year=2020)=>("Paper type"={Conference[8]})
2	6	(Year=2016)=>("Paper type"={Conference[6]})
3	6	(Year=2011)=>("Paper type"={Conference[6]})
4	4	(Year=2010)=>("Paper type"={Conference[4]})
5	3	(Year=2007)=>("Paper type"={Conference[3]})
6	2	(Year=2019)=>("Paper type"={Journal[2]})
7	1	(Year=2013)&(Reference=63)=>("Paper type"={Conference[1]})
8	1	(Year=2013)&(Reference=64)=>("Paper type"={Conference[1]})
9	1	(Year=2013)&(Reference=65)=>("Paper type"={Conference[1]})
10	1	(Year=2013)&(Reference=66)=>("Paper type"={Conference[1]})

FIGURE 4: Rules set supported.

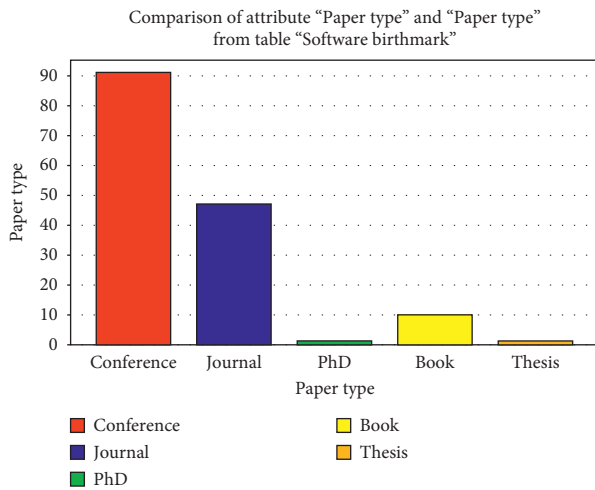


FIGURE 5: Frequencies of the dataset used.

intellectual software asset management [8], detection of software theft [18], plagiarism detection [38], detecting java theft [39], detecting binary theft [40], semantics-based repackaging detection for mobile apps [41], malware detection [42], detecting code theft [43], detecting the theft of natural language [44], credible, resilient, and scalable detection of software plagiarism using authority histograms [45], detecting plagiarized mobile apps [46], efficient similarity measurement technique of Windows software [47], detecting common modules in Java packages [48], measuring similarity of android applications [49], identify similar classes and major functionalities [50], moreover, for the source code level [48], and so on.

The proposed study has used the application of machine learning for software birthmark usability for transformation of source code. Initially, the K-nearest neighbors algorithm was used for classification of the software birthmarks. The experimental results of K-NN were effective and showed an accuracy of 98%. Figure 5 represents the frequencies of the dataset in term of conference, journals, books, and thesis.

Figure 6 shows the comparisons of the algorithms used in the proposed research. The algorithm decision rule has 0.91%, decomposition tree algorithm is having 0.96%, and the LTF-C algorithm is having 0.64% accuracy.

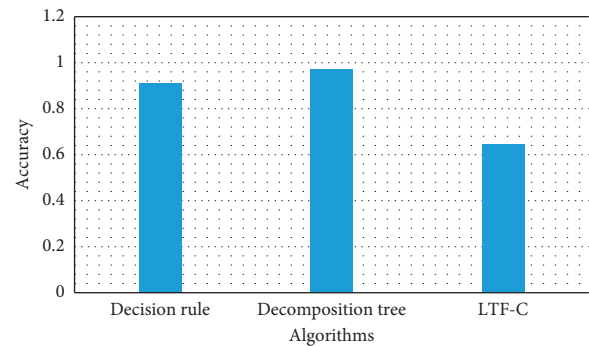


FIGURE 6: Algorithm used along with the accuracy.

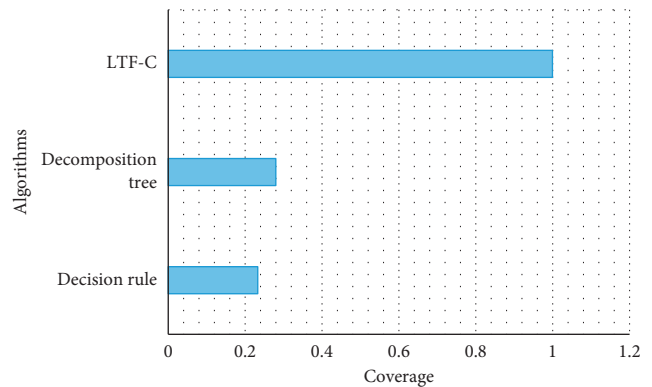


FIGURE 7: Coverage of the algorithms used.

Figure 7 graphically represents the coverage of the algorithms used.

5. Conclusion

Software industry is growing with the passage of time. New innovations are offered to cater diverse issues of real life. The role of software applications has evidenced the success of software industry. Pirates are engaged with code transformations and gaining profit from the code obfuscation, transformation of source code, and piracy of software. This is mostly carried out in situations of piracy where the pirates want the ownership of the software program. Various approaches are being practiced for source code transformation

and code obfuscation. Among the present approaches, software birthmark was one of the approaches developed with the aim to detect software piracy exists in the software. Birthmarks are considered to insist on the source code and executable of certain programming languages. The proposed study has used machine learning algorithms for classification of the usability of existing software birthmarks in terms of source code transformation. The K-nearest neighbors algorithm was used for classification of the software birthmarks. For cross-validation, the algorithms of decision rules, decomposition tree, and LTF-C were used. The experimental results show the effectiveness of the proposed research. The algorithm decision rule has 0.91%, decomposition tree algorithm is having 0.96%, and the LTF-C algorithm is having 0.64% accuracy.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was sponsored in part by the Research Fund for the Doctoral Program of Liaoning University of International Business and Economics (2019XJLXBSJJ002).

References

- [1] C. Gottschlich, "Curved-region-based ridge frequency estimation and curved gabor filters for fingerprint image enhancement," *IEEE Transactions on Image Processing*, vol. 21, no. 4, pp. 220–227, 2012.
- [2] C. S. Collberg, C. Thomborson, and G. M. Townsend, "Dynamic graph-based software fingerprinting," *ACM Transactions on Programming Languages and Systems*, vol. 29, no. 6, p. 35, 2007.
- [3] R. Thabit and B. E. Khoo, "Robust reversible watermarking scheme using Slantlet transform matrix," *Journal of Systems and Software*, vol. 88, pp. 74–86, 2014.
- [4] Y. Zeng, F. Liu, X. Luo, and C. Yang, "Software watermarking through obfuscated interpretation: implementation and analysis," *Journal of Multimedia*, vol. 6, no. 4, pp. 329–340, 2011.
- [5] C. Collberg and T. R. Sahoo, "Software watermarking in the frequency domain: implementation, analysis, and attacks," *Journal of Computer Security*, vol. 13, no. 5, pp. 721–755, 2005.
- [6] S. Nazir, S. Shahzad, R. Wirza et al., "Birthmark based identification of software piracy using Haar wavelet," *Mathematics and Computers in Simulation*, vol. 166, pp. 144–154, 2019.
- [7] T. Yokoi and H. Tamada, "A beforehand extraction method for dynamic software birthmarks using unit test codes," in *Proceedings of the 2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pp. 169–175, Busan, South Korea, June 2018.
- [8] D. Kim, J. Moon, S. J. Cho et al., "A birthmark-based method for intellectual software asset management," in *Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication*, Siem Reap, Cambodia, January 2014.
- [9] Z. Tian, Q. Zheng, T. Liu, and M. Fan, "DKISB: dynamic key instruction sequence birthmark for software plagiarism detection," in *Proceedings of the High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC)*, pp. 619–627, Zhangjiajie, China, November 2013.
- [10] L. Ma, Y. Wang, F. Liu, and L. Chen, "Instruction-words based software birthmark," in *Proceedings of the 2012 Fourth International Conference on Multimedia Information Networking and Security*, Nanjing, China, November 2012.
- [11] P. P. F. Chan, L. C. K. Hui, and S. M. Yiu, "JSBIRTH: dynamic JavaScript birthmark based on the run-time heap," in *Proceedings of the 2011 IEEE 35th Annual Computer Software and Applications Conference*, Munich, Germany, July 2011.
- [12] S. Nazir, S. Shahzad, and L. S. Riza, "Birthmark-based software classification using rough sets," *Arabian Journal for Science and Engineering*, vol. 42, no. 2, pp. 859–871, 2016.
- [13] S. Nazir, S. Shahzad, R. B. Atan, and H. Farman, "Estimation of software features based birthmark," *Cluster Computing-The Journal of Networks Software Tools and Applications*, vol. 21, no. 1, pp. 1–14, 2017.
- [14] S. Nazir, S. Shahzad, and N. Mukhtar, "Software birthmark design and estimation: a systematic literature review," *Arabian Journal for Science and Engineering*, vol. 44, no. 4, pp. 3905–3927, 2019.
- [15] H. Tamada, M. Nakamura, and A. Monden, "Design and evaluation of birthmarks for detecting theft of java programs," in *Proceedings of the IASTED International Conference on Software Engineering*, pp. 17–19, Innsbruck, Austria, February 2004.
- [16] G. Neufeld, "Descriptive name resolution," *Computer Networks and ISDN Systems*, vol. 23, no. 4, pp. 211–227, 1992.
- [17] G. Derrick, *Protection of Computer Software: Its Technology and Application*, p. 224, Cambridge University Press, New York, NY, USA, 1992.
- [18] G. Myles and C. Collberg, "Detecting software theft via whole program path birthmarks," in *Proceedings of the Information Security: 7th International Conference, ISC 2004*, Palo Alto, CA, USA, September, 2004.
- [19] E. H. Spafford and S. A. Weeber, "Software forensics: can we track code to its authors?" *Computers & Security*, vol. 12, no. 6, pp. 585–595, 1993.
- [20] S. Nazir, S. Shahzad, and S. B. S. Abid, "Selecting software design based on birthmark," *Life Science Journal*, vol. 11, no. 12, pp. 89–93, 2014.
- [21] S. Nazir, "Design and estimation of features based software birthmark," Ph. D. Thesis, University of Peshawar, Peshawar, Pakistan, 2015.
- [22] S. Nazir, S. Shahzad, S. A. Khan, N. B. Ilya, and S. Anwar, "A novel rules based approach for estimating software birthmark," *Scientific World Journal*, vol. 2015, Article ID 579390, 8 pages, 2015.
- [23] S. Nazir, S. Shahzad, Q. U. A. Nizamani, R. Amin, M. A. Shah, and A. Keerio, "Identifying software features as birthmark," *Sindh University Resarch Journal (Science Series)*, vol. 47, no. 3, pp. 535–540, 2015.
- [24] Y. Bai, X. Sun, G. Sun, X. Deng, and X. Zhou, "Dynamic k-gram based software birthmark," in *Proceedings of the 19th Australian Conference on Software Engineering*, Perth, Australia, March 2008.

- [25] H. Tamada, M. Nakamura, A. Monden, and K.-I. Matsumoto, "Detecting the theft of programs using birthmarks," Technical Report NAIIST-IS-TR2003014, Nara Institute of Science and Technology, Ikoma, Japan, 2003.
- [26] Y. Zeng, F. Liu, X. Luo, and S. Lian, "Abstract interpretation-based semantic framework for software birthmark," *Computers & Security*, vol. 31, no. 4, pp. 377–390, 2012.
- [27] D. Schuler, V. Dallmeier, and C. Lindig, "A dynamic birthmark for java," in *Proceedings of the Twenty-Second IEEE/ACM International Conference on Automated Software Engineering*, Atlanta, GA, USA, November 2007.
- [28] H. Tamada, K. Okamoto, M. Nakamura, A. Monden, and K.-I. Matsumoto, "Dynamic software birthmarks to detect the theft of windows applications," in *Proceedings of the International Symposium on Future Software Technology*, Xi'an, China, October 2004.
- [29] S. Choi, H. Park, H.-I. Lim, and T. Han, "A static birthmark of binary executables based on API call structure," in *Proceedings of the 12th Asian Computing Science Conference on Advances in Computer Science: Computer and Network Security*, Doha, Qatar, December 2007.
- [30] S. Kang, H. Shim, S.-J. Cho, M. Park, and S. Han, "A robust and efficient birthmark-based android application filtering system," in *Proceedings of the 2014 Conference on Research in Adaptive and Convergent Systems*, Towson, MD, USA, October 2014.
- [31] K. Liu, T. Zheng, and L. Wei, "A software birthmark based on system call and program data dependence," in *Proceedings of the 2014 11th Web Information System and Application Conference*, Tianjin, China, September 2014.
- [32] G. Myles and C. Collberg, "k-gram based software birthmarks," in *Proceedings of the 2005 ACM Symposium on Applied Computing*, Santa Fe, NM, USA, March 2005.
- [33] Z. Tian, Q. Zheng, T. Liu, M. Fan, E. Zhuang, and Z. Yang, "Software plagiarism detection with birthmarks based on dynamic key instruction sequences," *IEEE Transactions on Software Engineering*, vol. 41, no. 12, pp. 1217–1235, 2015.
- [34] Y. Mahmood, S. Sarwar, Z. Pervez, and H. F. Ahmed, "Method based static software birthmarks: a new approach to derogate software piracy," in *Proceedings of the Computer, Control and Communication, 2009. IC4 2009*, Karachi, Pakistan, February 2009.
- [35] Y. Wang, F. Liu, D. Gong, B. Lu, and S. Ma, "CHI based instruction-words based software birthmark selection," in *Proceedings of the Fourth International Conference on Multimedia Information Networking and Security*, Nanjing, China, November 2012.
- [36] Z. Tian, Q. Zheng, T. Liu, M. Fan, X. Zhang, and Z. Yang, "Plagiarism detection for multithreaded software based on thread-aware software birthmarks," in *Proceedings of the 22nd International Conference on Program Comprehension*, Hyderabad, India, June 2014.
- [37] X. Wang, Y.-C. Jhi, S. Zhu, and P. Liu, "Detecting software theft via system call based birthmarks," in *Proceedings of the Computer Security Applications Conference*, pp. 149–158, Honolulu, HI, USA, December 2009.
- [38] D.-K. Chae, S.-W. Kim, J. Ha, S.-C. Lee, and G. Woo, "Software plagiarism detection via the static API call frequency birthmark," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, Coimbra, Portugal, March 2013.
- [39] H. Park, S. Choi, H.-I. Lim, and T. Han, "Detecting java theft based on static API trace birthmark," in *Proceedings of the Advances in Information and Computer Security, Third International Workshop on Security, IWSEC 2008*, Kagawa, Japan, November 2008.
- [40] S. Park, H. Kim, J. Kim, and H. Han, "Detecting binary theft via static major-path birthmarks," in *Proceedings of the 2014 Conference on Research in Adaptive and Convergent Systems*, Towson, MD, USA, October 2014.
- [41] Q. Guan, H. Huang, W. Luo, and S. Zhu, "Semantics-based repackaging detection for mobile apps," in *Proceedings of the Engineering Secure Software and Systems: 8th International Symposium, ESSoS 2016*, London, UK, April 2016.
- [42] S. Vemparala, F. D. Troia, V. A. Corrado, T. H. Austin, and M. Stamo, "Malware detection using dynamic birthmarks," in *Proceedings of the 2016 ACM on International Workshop on Security and Privacy Analytics*, New Orleans, LA, USA, March 2016.
- [43] H. Park, S. Seokwoo Choi, H.-I. Lim, and T. Taisook Han, "Detecting code theft via a static instruction trace birthmark for Java methods," in *Proceedings of the 2008 6th IEEE International Conference on Industrial Informatics*, pp. 551–556, Daejeon, South Korea, July 2008.
- [44] J. Yang, J. Wang, and D. Li, "Detecting the theft of natural language text using birthmark," in *Proceedings of the 2006 International Conference on Intelligent Information Hiding and Multimedia*, Pasadena, CA, USA, December 2006.
- [45] D.-K. Chae, J. Ha, S.-W. Kim, B. Kang, E. G. Im, and S. Park, "Credible, resilient, and scalable detection of software plagiarism using authority histograms," *Knowledge-Based Systems*, vol. 95, pp. 114–124, 2016.
- [46] D. Kim, A. Gokhale, V. Ganapathy, and A. Srivastava, "Detecting plagiarized mobile apps using API birthmarks," *Automated Software Engineering*, vol. 23, no. 4, pp. 591–618, 2015.
- [47] P. Daeshin, J. Hyunho, P. Youngsu, and H. JiMan, "Efficient similarity measurement technique of windows software using dynamic birthmark based on API," *Smart Media Journal*, vol. 4, no. 2, pp. 34–45, 2014, <http://KJD:ART002028547>, in Korean.
- [48] H. Park, H.-I. Lim, S. Choi, and T. Han, "Detecting common modules in java packages based on static object trace birthmark," *The Computer Journal*, vol. 54, no. 1, pp. 108–124, 2009, in English.
- [49] J. Ko, H. Shim, D. Kim et al., "Measuring similarity of android applications via reversing and k-gram birthmarking," in *Proceedings of the 2013 Research in Adaptive and Convergent Systems*, Montreal, Canada, October 2013.
- [50] T. Kakimoto, A. Monden, Y. Kamei, H. Tamada, M. Tsunoda, and K.-i. Matsumoto, "Using software birthmarks to identify similar classes and major functionalities," in *Proceedings of the 2006 International Workshop on Mining Software Repositories*, Shanghai, China, May 2006.

Research Article

Applying Code Transform Model to Newly Generated Program for Improving Execution Performance

Bao Rong Chang ¹, Hsiu-Fen Tsai ², and Po-Wen Su ¹

¹Department of Computer Science and Information Engineering, National University of Kaohsiung, Kaohsiung, Taiwan

²Department of Fragrance and Cosmetic Science, Kaohsiung Medical University, Kaohsiung, Taiwan

Correspondence should be addressed to Hsiu-Fen Tsai; sftsai@kmu.edu.tw

Received 20 November 2020; Revised 17 January 2021; Accepted 20 January 2021; Published 2 February 2021

Academic Editor: Sikandar Ali

Copyright © 2021 Bao Rong Chang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The existing programs inside the voice assistant machine prompt human-machine interaction in response to a request from a user. However, the crucial problem is that the machine often may not give a proper answer to the user or cannot work out the existing program execution efficiently. Therefore, this study proposes a novel transform method to replace the existing programs (called sample programs in this paper) inside the machine with newly generated programs through code transform model GPT-2 that can reasonably solve the problem mentioned above. In essence, this paper introduces a theoretical estimation in statistics to infer at least a number of generated programs as required so as to guarantee that the best one can be found within them. In addition, the proposed approach not only imitates a voice assistant system with filtering redundant keywords or adding new keywords to complete keyword retrieval in semantic database but also checks code similarity and verifies the conformity of the executive outputs between sample programs and newly generated programs. According to code checking and program output verification, the processes can expedite transform operations efficiently by removing the redundant generated programs and finding the best-performing generated program. As a result, the newly generated programs outperform the sample programs because the proposed approach reduces the number of code lines by 32.71% and lowers the program execution time by 24.34%, which is of great significance.

1. Introduction

Alpha Go was developed by Google DeepMind in London in 2014, and it defeated all other Go masters. Since then, research in artificial intelligence has been increasing again. Accordingly, research about human-computer interaction is meant to imitate human behavior, especially natural language representation and interpretation in the voice assistant machine [1]. The use of artificial intelligence for human-computer interaction in voice assistant machines-related tools have flourished, such as Tesla's NoA, Apple's Siri, Amazon Echo and Alexa, and Google Home. These tools not only function as the basis of human language imitation but also play a key role for API offerings in AI applications. Nowadays, there are quite a lot of brands and types of voice assistant machines in the world, and their

existing programs are used for human-computer interaction in response to user requests. However, some voice assistant machines have encountered certain problems, that is, the machine may not be able to answer the questions correctly [2], or the existing programs have low execution efficiency [3]. It is worth exploring as to how to solve the above problems, and thus we have formulated the following idea. Can the machine transform an existing program to a newly generated program that can run programs with higher efficiency and produce correct results?

Even though the deep learning model is relatively mature, imitation of human natural language behavior through deep learning is still a difficult task. Language model is a technology that allows machines to understand and predict human language. The use of recently developed language models with large-scale data and huge computing power can help solve

various applications of human natural language. Famous open-source pretrained language models, such as ELMo, BERT, and GPT-2 [4], can implement the best level of Natural Language Processing (NLP) tasks, and they have the ability required for huge hierarchical models. Through transfer learning and few-shot learning on such powerful natural language models, the solution to the complex NLP problem can be obtained.

The objective of this study was to explore ways to solve the following two problems. First, how to construct a complete semantic database that can implement data retrieval quickly and respond correctly. Second, how to generate high-performance programs through the code transform model to replace the existing programs inside the machine to improve execution performance. The natural language processing involves understanding and generating. Regarding the technology involved in the improvement method, a complete and quickly searchable semantic database using MariaDB is constructed with the Natural Language Toolkit (NLTK) model of sentence segmentation [5] to provide correct answers to users. At present, the most fluent open source natural language model is the Generative Pre-trained Transformer 2 (GPT-2) [6], which is a natural language simulation machine developed by using OpenAI. It is currently being used to generate fake news [7]. In this study, Python [8] provides the run-time environment for the aforementioned tools, namely, MariaDB, NLTK, GPT-2, and Tensorflow. Python combined with Hadoop Streaming to provide big data processing and distributed storage in Hadoop, and it can also be used with PySpark to provide big data analytics in Spark.

There is so far no way to infer how many programs generated from GPT-2 can guarantee a pass ratio over 90% without which we cannot find the best one among them. Nevertheless, this paper introduces a theoretical estimation in statistics to infer at least a number of generated programs produced by GPT-2, which guarantee that the best one can be found within them. This approach refers to the pre-determined the number of generated programs statistically. In order to improve the efficiency of keyword retrieval, both the parameters of filtering redundant keywords and adding new keywords are used to optimize keyword-search in the semantic database in order to improve the hit rate of keywords from the database. In addition, we present cosine text similarity algorithm Simhash [9] to check the code similarity of generated programs and example programs and use the longest common subsequence (LCS) [10] to verify the conformity between the execution results of generated programs and example programs.

The following paragraphs of this paper are arranged as follows. In Section 2, related work in word segmentation processing and language generation model will be described. The way to system implementation is given in Section 3. The experimental results and discussion is given in Section 4. Finally, we drew a brief conclusion in Section 5.

2. Related Work

A particular voice assistant machine, Amazon Alexa [11], is a smart assistant and Echo smart speaker developed in recent years. The audio system relies on a set of complex AI

technologies that use automatic speech recognition (ASR) to receive sound waves and to convert them into words. Then, natural language understanding (NLU) is used to translate these words into meanings. In order to respond to people, smart speakers and the technology of natural language generation (NLG) [12] for suggestions, content determination, conversation planning, sentence aggregation, vocabulary, citation expression generation, language experiment, and descriptions of basic NLG task have been developed. While exploring various emerging issues of AI, some researchers are pondering the possibility of machines automatically producing programs [13]. In reality, people are often pressurized to complete programming or modify the codes of a program before the deadline or the challenge of improving the execution of a program. As a result, the program execution speed may be too slow to get the job done, and there may still be deficiencies when running the program in real-time. In contrast, a high-performing new program can be generated through the code transform model, and it will operate faster than the original one [14]. This would be a great progress in the application of AI, and the audio-text conversion technology would further apply in the sound-controlled related products, such as drones, robots, and flying iron men.

There is not much literature on source-code to source-code transform model. First, Li et al. (2011) used XML and XSLT technology to generate a web page code [15]. They will be automatically converted into target code after being exported via XML files. However, they did not provide any data to show how the results were generated. Next, Li et al. (2020) employed Java code transform models, Java-Codetool, and CodeGeneration to produce newly generated java programs and evaluate their performance [16]. They took the Binary tree traversal program as an example and divided the program into 4 parts to produce newly generated source-code programs individually. The experimental results show that the average time to generate a code line in a program using Java-Codetool is about 0.17 seconds, and CodeGeneration takes 0.15 seconds. No matter whether it is using Java-Codetool or CodeGeneration, the number of code lines is not reduced. In fact, the compilation of all generated programs can be done successfully. However, it did not show further information about the execution results of their generated programs. In this paper, the proposed approach will present the code similarity check and make sure the conformity of the execution results between generated programs and sample programs that resulted in the credibility and validity of the findings in this study.

Uncertainties will exist depending on the situation. Despite this, the system can generate similar programs instantly to greatly reduce the possibility of errors or incomplete programs with GPT-2. In order to realize the “applying code transform model to newly generated program for improving execution performance”, this study will use the following key technologies: Anaconda (Data Science Platform with Virtual Environment Conda), Tensorflow (Dataflow and Differentiable Programming), NLTK (English Text Segmentation), GPT-2 (Text Generating Model), and

Simhash (Cosine Text Similarity Algorithm), etc., to achieve the goal of this research.

2.1. Word Segmentation Processing-NLTK. Natural Language Processing (NLP) [17] is regarded as a branch of AI and linguistics. This field discusses how to process and use natural language, including multiple steps, basically for cognition, understanding, generation; cognition and understanding are to let the computer turn the natural language input into interesting symbols and relationships. Natural Language Toolkit (NLTK) is a Python library commonly used in NLP research. This is an open source project, including data sets, Python modules, tutorials, etc.

The main functions of NLTK are English word segmentation processing, part-of-speech tagging [18] called pos, font restoration called lemmatization, stopword, named entity recognition called ner, etc. In NLTK, the text is usually stored as a list, that is, a text is a huge list. If additional information such as part-of-speech is attached, it can be converted into a dictionary. The Latin language system is a little troublesome because they like to add a modifier to the words to describe different tenses, actions, part-of-speech, mood, and quantity. So we will consider all the same words in different tenses or different changes into the same word for processing. Finally, we filter out unnecessary words. and the English word segmentation flowchart is shown in Figure 1.

2.2. Language Generation Model-Generative Pre-Training 2. The second generation of Generative Pre-Training (GPT-2) is an unsupervised [19] transformer language generation model [20], released by OpenAI in 2019. Researchers believe that the language model of unsupervised learning is a general language model. Furthermore, GPT-2 proves that the model is not meant for any specific task to predict the next word as the training target. It is used to adopt the sentence database WebTex [21] for data training, which contains 8 million web pages as the training data. These web pages are part of the data from Reddit [22] and are more than 40 GB. Compared with other text-generating models, such as ELMo and Bert for producing texts, its main advantage is that the code is in English and the one-way language model is easier to train and understand. The traditional Transformer model is composed of Encoder and Decoder, called the Transformer architecture stack. The transformer architecture stack is shown in Figure 2. This model solves the problem of machine translation.

In many subsequent studies, the Transformer architecture removes either the Encoder or the Decoder, uses only one Transformer stack, and provides a large amount of training text and machine equipment. GPT-2 is composed of the Decoder architecture according to the Transformer model. As shown in Figure 3, the stacking height is the size of various GPT-2 models. Currently, there are four sizes of models: GPT-2 Small, GPT-2 Medium, GPT-2 Large, and GPT-2 Extra Large [23].

2.3. Cosine Text Similarity Algorithm-Simhash. The traditional Hash [9] algorithm is only responsible for mapping the original content into a signature value equally and randomly. The principle is only equivalent to a pseudo-random number generating algorithm. Even if the two original contents differ in only one byte, the generated signature value may be very different. Therefore, traditional Hash cannot measure the similarity of the original content in the dimension of signature. Simhash algorithm [24] is a locally sensitive Hash algorithm, and its main idea is to reduce the dimensionality of feature vector. That is, Simhash algorithm is used to convert the high-dimensional feature vector into f-bit fingerprint [25] and to determine the similarity of two f-bit fingerprints by calculating the Hamming distance [26] of these fingerprints. The smaller the Hamming distance, the higher the similarity. The overall process is shown in Figure 4, which includes word segmentation, hash calculation, weighting, merging, and dimensionality reduction. Word segmentation is used to obtain N-dimensional feature vectors (64-dimensional default) for the word segmentation of the text; Hash is to perform the Hash calculation on all the obtained feature vectors. Weighting refers to weighing all the obtained feature vectors. Merging refers to accumulation of all the obtained vectors. Dimensional reduction changes the accumulated result greater than zero to one and less than zero to zero, obtains a text Fingerprint as shown in Figure 5, and finally calculates the Hamming distance between the two text fingerprints.

In the Information Theory, the Hamming distance between two equal-length character strings is the number of different characters at the positions corresponding to the two characters. The Hamming distance is the number of characters that need to be replaced to convert one string into another for a fixed length. Moreover, Hamming distance is a distance measure for the character vector space, and it maintains the measured distance with nonnegative, unique, and symmetrical. In Hamming distance formula equation (1) [27], d^{HAD} is the Hamming distance between objects i and j , and k is the index of the corresponding variable reading y in the total number of variables n . In equation (2) and equation (3), $[y_{i,k} \neq y_{j,k}]$ is the value of 1 or 0 given by the logical value True or False determined according to internal conditions $y_{i,k} \neq y_{j,k}$. The Hamming distance itself gives the number of mismatches between variables paired by k .

$$d^{\text{HAD}} = \sum_{k=0}^{n-1} [y_{i,k} \neq y_{j,k}], \quad (1)$$

$$[y_{i,k} \neq y_{j,k}] = 1 \text{ if } y_{i,k} \neq y_{j,k} \text{ is True,} \quad (2)$$

$$[y_{i,k} \neq y_{j,k}] = 0 \text{ if } y_{i,k} \neq y_{j,k} \text{ is False.} \quad (3)$$

If the Hamming distance is used to measure the similarity of the original content, the similarity can be converted into a pass ratio as a measure that tests the object based on

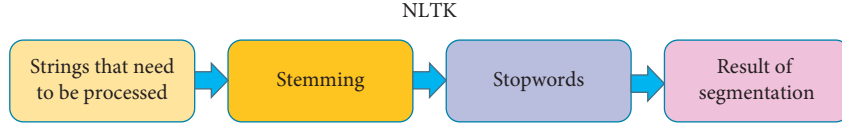


FIGURE 1: NLTK word segmentation process.

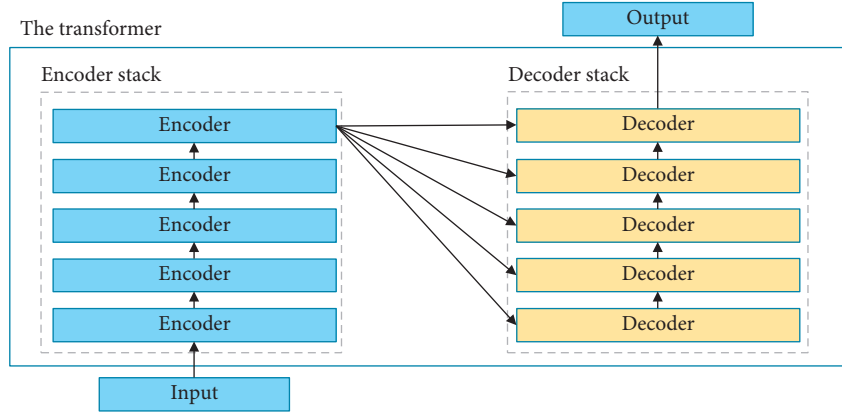


FIGURE 2: Transformer architecture stack.

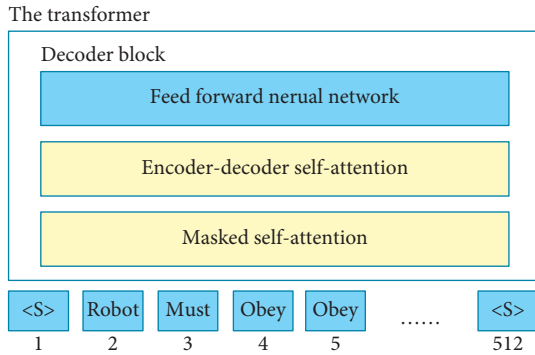


FIGURE 3: GPT-2 Decoder architecture.

the original standard. According to the Hamming distance d^{HAD} and the total number of variables n , the qualification ratio formula equation (4) can be obtained.

$$q = \left(1 - \frac{d^{\text{HAD}}}{n}\right) \times 100\%. \quad (4)$$

2.4. Longest Common Subsequence (LCS). The Longest Common Subsequence [10], abbreviated LCS, is the problem of finding the longest common subsequence in all sequences in a sequence set (usually two sequences). This is different from the problem of finding the longest common substring (Longest Common Substring) in that the subsequence does not need to occupy consecutive positions in the original sequence. To solve the LCS problem, we cannot use the brute force search method. We need to use dynamic programming to find the length of the LCS and backtracking strategy to find the actual sequence of the LCS.

We assume that $z = \langle z_1, z_2, \dots, z_k \rangle$ is the LCS of x and y , and we observe that if $x_m = y_n$, then $z_k = x_m = y_n$, and z_{k-1} is the LCS of x_{m-1} and y_{n-1} ; If $x_m \neq y_n$, then z_k is the LCS of x_{m-1} and y_{n-1} , or the LCS of x_{m-1} and y_n . Therefore, the problem of solving LCS becomes two sub-problems of a recursive solution. However, in the abovementioned recursive solution method, there are many repeated sub-problems and the efficiency is low. The improved method uses space instead of time and uses an array to store intermediate states to facilitate subsequent calculations. Therefore, using the two-dimensional array $c[i, j]$ to record the LCS lengths of the strings x_1, x_2, \dots, x_i and y_1, y_2, \dots, y_j , the state transition equation can be obtained in equation (5).

$$c[i, j] = \begin{cases} 0, & i = 0 \text{ or } j = 0, \\ c[i - 1, j - 1] + 1, & i, j > 0 \text{ and } x_i = y_j, \\ \max(c[i, j - 1], c[i - 1, j]), & i, j > 0 \text{ and } x_i \neq y_j. \end{cases} \quad (5)$$

The longest common subsequence is used to measure the similarity of the execution results of two programs [28]. In order to illustrate the degree of conformity of the respective output results of the two programs, the similarity is renamed as text conformity. First, convert the output results of individual programs into ASCII code, and then store them into arrays a and b individually, and then calculate the c array according to the longest common subsequence. Here, the lengths of the a , b , and c arrays are recorded as $|a|$, $|b|$ and $|c|$, and the length of the above array is substituted into the formula equation (6) to obtain the text conformity. Here, the lengths of arrays a , b , and c are denoted as $|a|$, $|b|$, and $|c|$, and the length of the above array is substituted into equation (6) to obtain LCS conformity, denote as f .

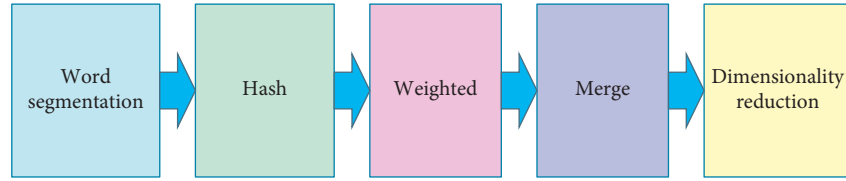


FIGURE 4: Simhash algorithm process flow.

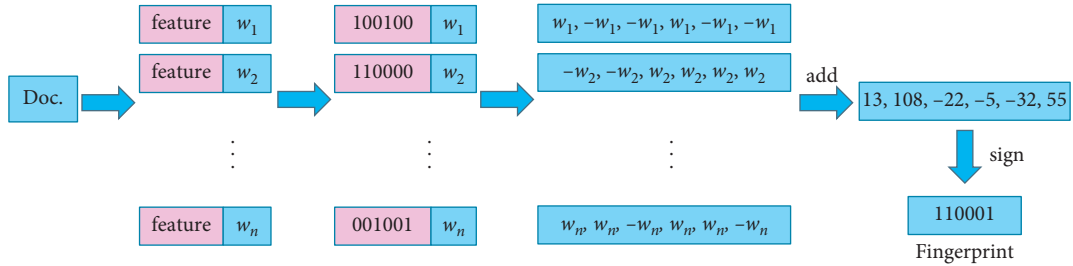


FIGURE 5: Calculate fingerprint.

$$f = \frac{2 \cdot |c|}{|a| + |b|} \times 100\%. \quad (6)$$

3. Research Method

The purpose of this study was to produce newly generated programs for improving program execution speed using the code transform mode. In other words, in order to increase the efficiency of human-machine interaction like voice assistant machine, the existing programs (called sample programs in this paper) inside the machine is replaced with a new high-performance program using GPT-2. Nevertheless, the proposed approach, as shown in Figure 6, consists of three parts: prior keyword retrieval optimization, code transform model, and posterior verification of generated program. The first part is to imitate a voice assistant machine to segment a sentence, select keywords, and find sample programs. Next, a transform model is used to produce a number of new candidate programs according to the corresponding sample programs. The last one is to test and verify the new candidate programs and choose the one with the best quality as a pocket program. A pocket program is the one with the best performance among new candidate programs.

3.1. Newly Generated Program System. The objective of this paper is to explore the use of the second generation of Generative Pre-Training (GPT-2) as a code transform model to produce newly generated high-performance programs. In this way, the sample program in the semantic database can be replaced with a new program called the pocket program to implement high-performance execution due to reduced code lines and less execution time. With GPT-2 transform model, the proposed approach has to imitate a voice assistant system with prior optimization of keyword retrieval associated with the semantic database, and to build a content checker using posterior verification of program execution results between a sample program and the newly generated

program. The system includes model generation stage and model use stage, as shown in Figure 7. In the model generation stage, the user sends spoken sentences using text or voice and then proceeds with word segmentation to select keywords and find corresponding sample programs. After that, the system trains the model and generates preliminary programs in which some of the programs are chosen as qualified programs. Finally, a generative program model is confirmed and saved in the semantic database. During the model use stage, the user sends a spoken sentence in text or voice, and searches the semantic database for pocket programs that could be made earlier in the model generation stage. If not, it will go back to the step of generating preliminary programs, and then go through test and verification steps to get a new pocket program for execution.

The model generation stage is divided into the training phase and the test phase. Phase diagrams are shown in Figures 8–10. In the training stage, there are four units: word segmentation unit, sample program unit, generative program model unit, and the generated program unit. The inputs/outputs during training phase are natural language sentences, keywords, sample programs, generative program models, and preliminary programs. Natural language sentences are initially sent by a user. The word segmentation model NLTK is used to perform word segmentation. The system is used to select the keywords and then search the semantic database for the sample program that is corresponding to the keywords. Semantic database is built in the XAMPP [29] cloud server. System puts the sample program into GPT-2 for the first pass. GPT-2 uses a sample program to train and generate the generative program model. After modeling is completed, the model will be provided as feedback to GPT-2 again for the second pass on which a number of preliminary programs are produced. In the test phase, there are three units: test unit, verification unit, and the storage unit. Simhash algorithm is used to compare the similarity between the preliminary program and the sample program. Qualified programs are obtained as the preliminary program with the similarity pass ratio higher than the

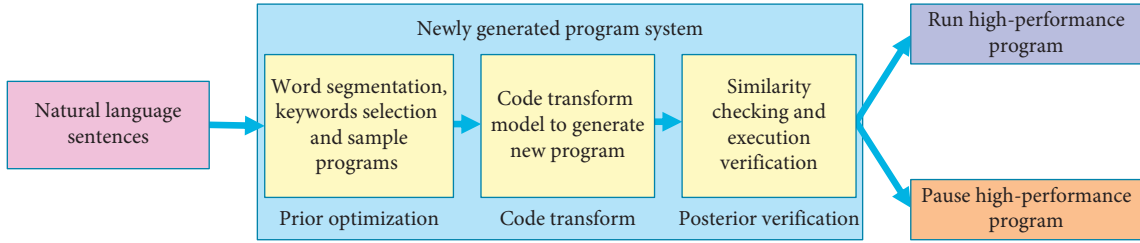


FIGURE 6: Applying the code transform model to the newly generated high-performance programs.

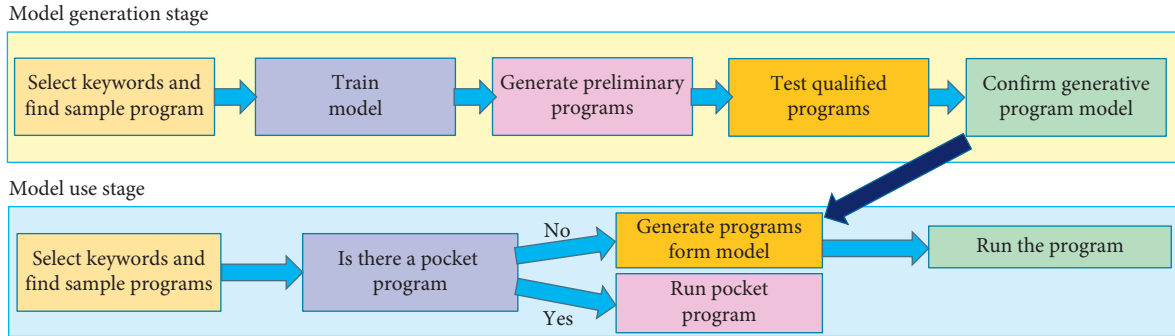


FIGURE 7: Diagram of newly generated program system.

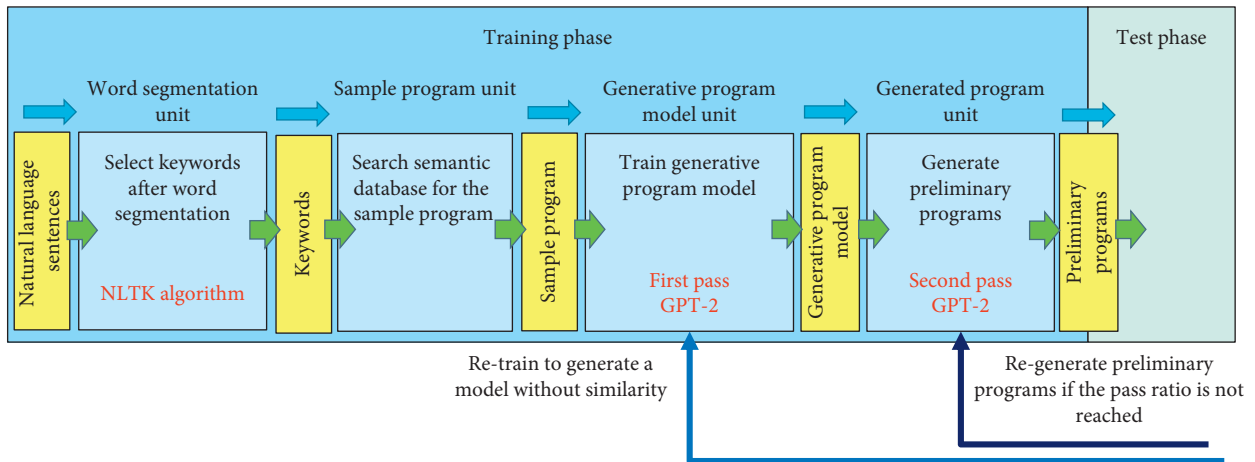


FIGURE 8: Model generation stage-training phase diagram.

predetermined one, and they are compiled with Python. Next, in the verification unit, we select the qualified program with the highest similarity pass ratio to compare with the corresponding sample program using LCS conformity that measures the conformance of their execution results, and chooses a qualified program that meets the predetermined conformity ratio as a pocket program. Finally, in the storage unit, the keywords, pocket programs, and generated program models will be stored in the semantic database.

With the trained generative program model found in semantic database, the corresponding selected keyword will be directed to the process of model use stage after the word segmentation. The stage diagram is shown in Figure 10. In the model use stage, there are five units: word segmentation unit, generative program model unit, generated program unit, test and verification unit, and storage unit. Likewise,

the word segmentation unit initially allows the user to send natural language sentences and use NLTK algorithm to segment a sentence and select keywords. Then, it will use keywords to search the semantic database for generative program models or pocket programs. If pocket programs are found, they will be executed directly. If not, it will go back to the step of generating preliminary programs, and then go through test and verification steps to get a new pocket program for execution.

3.2. *System Execution Flow.* Figure 11 shows the overall execution flow of the proposed approach. In the model generation stage, the user sends sentences or articles into the word segmentation model NLTK to perform word segmentation and select keywords from all separated words.

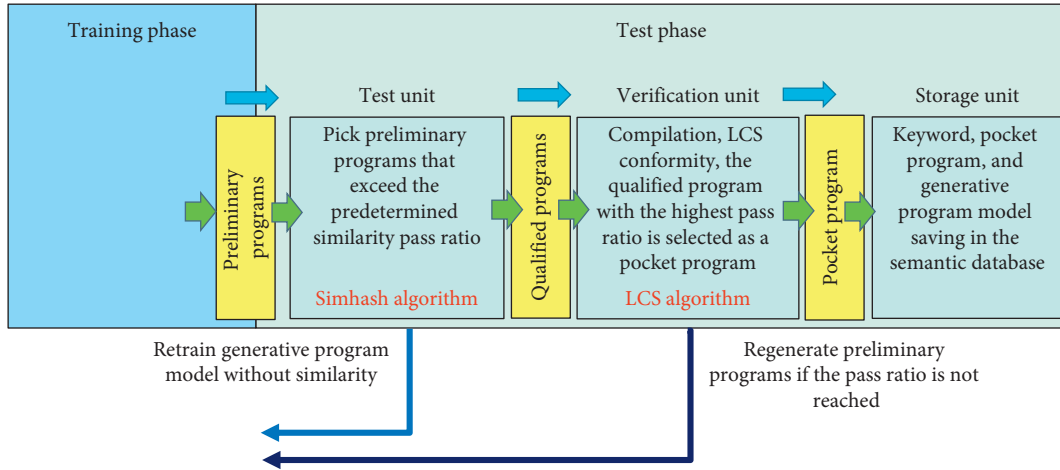


FIGURE 9: Model generation stage-test phase diagram.

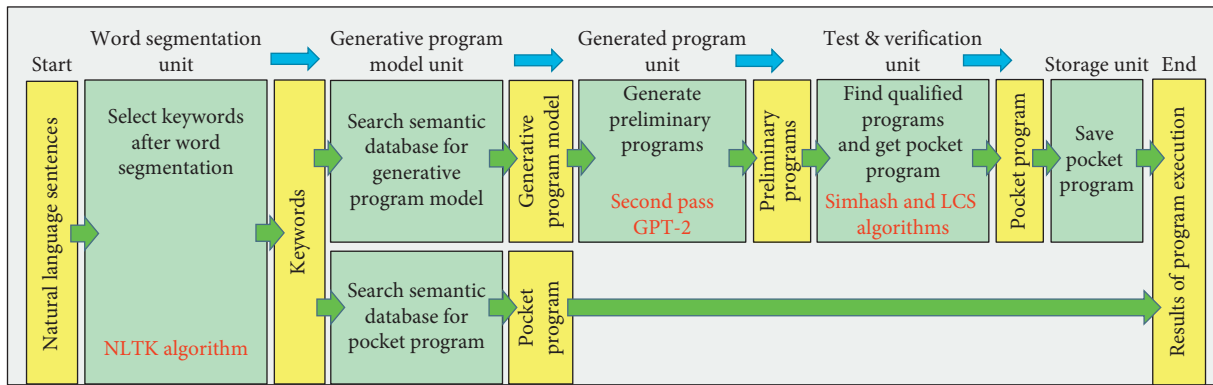
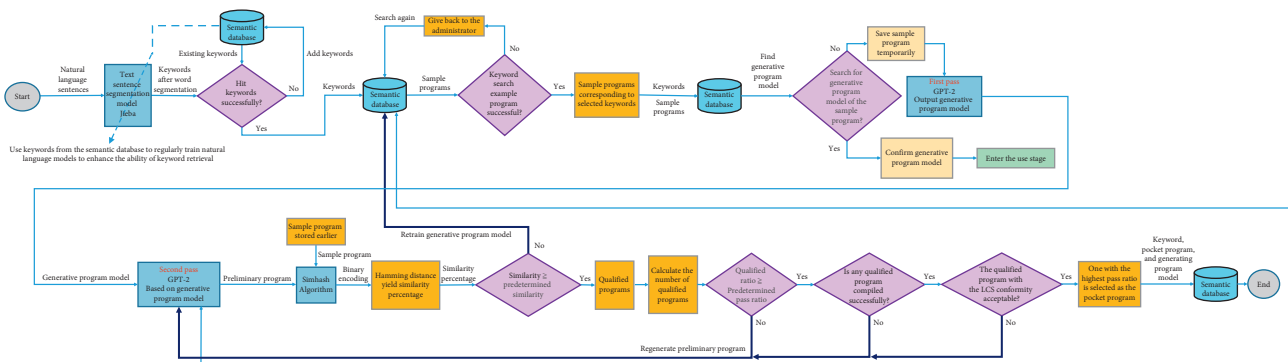


FIGURE 10: Model use stage diagram.

Execution flow of model generation stage



Execution flow of model use stage

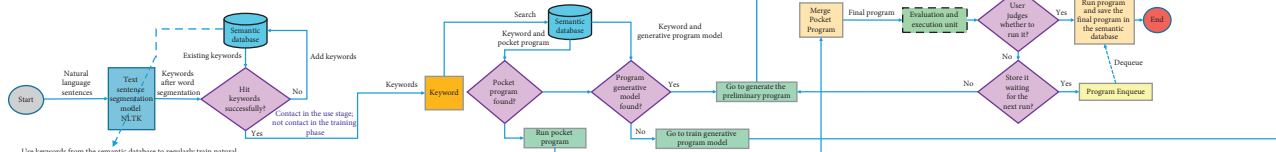


FIGURE 11: The overall execution flow of the program generation system.

Then, the selected keywords are checked against the semantic database to check if the same keywords exist. If not, the keywords are added to the semantic database. After the keywords are presented, the sample program path in the same row of the keywords in a table is obtained and judged. If there is a sample program path, we output the sample program according to the path to the next step. If there is no sample program, it is expected that a new sample program can be obtained through the web crawler to collect the proper sample program and store it into the semantic database. The procedure is then going to search for whether there is already a trained generative program model. If not, the sample program is put into GPT-2 as the first pass to produce the generative program model. If yes, the second pass is taken to feed the generative program model to GPT-2 so as to decode the model and generate 100 preliminary programs from GPT-2. The preliminary programs have been brought to the next test and verification steps. In the beginning, the 100 preliminary programs are sent individually to compare with the corresponding sample program and both are calculated with the Simhash signature value. Moreover, they are compared using Hamming Distance, and the similarity ratio is judged by the distance. After checking the code similarity between any one preliminary program and the sample program, this preliminary program will be viewed as the qualified program if its similarity exceeds the predetermined pass ratio (e.g., $\geq 90\%$). If so, it then will be sent to the next step for the verification of its execution result. However, if no preliminary program exceeds the predetermined pass ratio, it has to get back to the previous step to retrain a new generative program model and then try to regenerate 100 preliminary programs. Once all qualified programs have been produced, we need to check whether the number of qualified programs produced is enough. If there are only fewer qualified programs produced, it goes back to the previous step to re-generate a model. If the ratio of the number of qualified programs to 100 preliminary programs is big enough (e.g., $\geq 80\%$), the qualified programs are a majority and they are naturally compiled with Python. After the compilation is successful, LCS conformity between the execution result of the qualified program with the highest similarity pass ratio and the execution result of the corresponding sample program is computed to check whether their conformance meets the predetermined conformity (e.g., $\geq 95\%$). If so, such a qualified program serves as a pocket program. Finally, the pocket program, the generative program model, the keyword, and sample program are stored into the same row in a specific table in the semantic database.

In the model use stage, NLTK not only applies to segment words but also implements keyword drop-out and addin to optimize keyword retrieval. After the word segmentation, the useless keywords are not selected and the accuracy of the keyword hit is improved. Similarly, new keywords are added to the semantic database to improve the accuracy of the keyword hit. Next, we will check if a corresponding pocket program in the semantic database has been executed before. If the corresponding pocket program exists, the pocket program will be provided to the user for

execution. On the other hand, if the corresponding pocket program does not exist, it moves on to the step of training a new generative program model. After that, the subsequent step is to pick up several pocket programs corresponding to the respective keywords in the database and then merge them together to form a complete final program. "Evaluation and execution unit" is a further step to carry out a final program. It is expected that we can evaluate the execution performance of the final program to see how long it will take. In this way, the user may allow it to be taken or aborted.

3.3. Hardware Specification and Recipe of Software Tools.

In the model training phase, a high-level GPU cluster used for GPT-2 execution is used for rapid model training to reduce the processing time spent on traditional CPU. In Table 1, the operation would be carried out by the following tools: (1) NLTK word segmentation model, (2) unsupervised Generative Pre-Training second-generation transformer language model, (3) Simhash algorithm as an advanced version of Hash algorithm, and (4) LCS algorithm as an advanced version of DP algorithm.

The hardware equipment for running the program is based on two Nvidia-brand GPU P100 and two RTX2080Ti. Four GPU cluster workstations are connected through a high-speed local network to accelerate the calculation [30]. Cluster workstations have higher availability, reliability, and scalability than a workstation. Each workstation server transmits data through the high-speed network QPI, and uses a hardware interface PCIe x16 channel to connect both CPU and GPU. The GPU link uses NVLink [31] developed by Nvidia to allow four GPUs to share memory by using a point-to-point structure and serial transmission. Not only between CPU and GPU, but the connection between multiple Nvidia GPUs are also established. Under multiple GPUs, SLI, Surround, and PhysX options will show in the Nvidia system panel. Turning on the SLI, the users can share the graphics card memory for more data calculation. The detail hardware specifications are shown in Table 2. The overall architecture diagram is shown in Figure 12.

3.4. Evaluation of the Performance of Keyword Retrieval.

An experiment of the feasibility of generating the program model was conducted in two parts: keyword selection from a sentence and keyword retrieval optimization [32], and the estimation of the number of programs generated from code transform model GPT-2 and the calculation of the pass ratio of similarity checking between a generated program and a sample program. The first part is used to optimize the keyword-searching in a semantic database in order to improve the success of hit keywords from the database. There are two ways to optimize keyword-searching. The first one is to filter the redundant keywords in a sentence after word segmentation, and the second one is to add the new keywords into the semantic database so as to increase the success of hit keyword hits from the database. The first one filters keywords to select the keywords that were separated from the NLTK word segmentation. The nonrelated connectives or auxiliary words are deleted to improve the

TABLE 1: Open-source software tools.

Package	Version
Anaconda2	5.2.0
Python	3.7.5
Tensorflow	1.14
CUDA	10
XAMPP	3.2.4
NLTK	3.5
GPT-2	0.6
SimHash	2.0.0
LCS	Unknown

TABLE 2: Hardware specification.

Hardware	Specification	Amount
Server	HP Z8 G4 workstation	2
	HP Z4 G4 workstation	2
CPU	Xeon silver 4108	4
	I9-7900X	2
Ram	DDR4-2666 8G	16
	DDR4-2666 16G	12
Disk	MDFDDAK512TBN-1AR1ZABHA	2
	SAMSUNG-MZVPV256HEGL	2
	TOSHIBA-DT01ACA200	2
GPU	NVIDIA Quadro GP100	2
	NVIDIA GeForce RTX 2080 Ti 11G	2
Network	Intel ethernet connection X722 for 1GbE	4

accuracy of hit keywords from the semantic database. The second one is to add keywords to check if the necessary keywords exist in the semantic database, to add keywords to the semantic database, and to notify the users to find and add the corresponding sample programs in the database as well. To improve the hit ratio of keyword retrieval if the user submits the same keyword again, it is necessary to check whether the hit ratio of keyword retrieval in the semantic database is increased. Because F1-Score [33] is often used as a measure of accuracy in pattern recognition, sample survey, and information retrieval, this study used F1-Score as an Evaluation Metric to measure the accuracy. F1-Score is a harmonic average calculated using Precision and Recall. To find the F1-Score, the user must firstly define the positive class, negative class, and consider whether it is retrievable. The definition of the Confusion Matrix [34] is shown in Table 3.

In the confusion matrix about the keyword-searching issue, there are four terms. True positive (TP) is positive and judged to be positive. False positive (FP) is negative but wrongly judged to be positive. False negative (FN) is positive but wrongly judged to be negative. True negative (TN) is negative and judged to be negative. The accuracy is defined in equation (7), which is to calculate the ratio of TP to TP+FP. The recall is defined in equation (8), which is to calculate the ratio of TP to TP+FN. The recall is defined in equation (8), which meant is to calculate the proportion of all results (TP + FN) retrieved by the positive results (TP). After finding the accuracy and recall, F1-Score is the harmonic average of accuracy and recall. It is defined in equation (9). After adjusting equation (9), F1-Score is

concise in equation (10). Corresponding to the F1-Score evaluated after word segmentation, the default keywords are selected to check the performance of keyword retrieval from the semantic database. TP refers to those keywords that are retrieved and positively related. The nonrelevant retrieved keywords are FP. The nonretrieved but related keywords are FN. For evaluation metrics, the process of calculating accuracy, recall, and F1-Score is conducted.

$$P = \frac{TP}{TP + FP}, \quad (7)$$

$$R = \frac{TP}{TP + FN}, \quad (8)$$

$$\frac{2}{F1} = \frac{1}{P} + \frac{1}{R}, \quad (9)$$

$$F1 = \frac{2 \cdot P \cdot R}{P + R} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}. \quad (10)$$

3.5. Performance Evaluation of Program Execution. This section evaluates the performance improvement of newly generated programs produced by GPT-2. In other words, we are going to compare the execution performance between sample program and each generated program individually. The performance evaluation includes (1) comparing the number of code lines of the sample program and the average number of code lines of generated programs and (2) comparing the execution time between them as well. Two indicators are used to explain how much performance would be improved for any program execution where the first one is to measure the ratio of average code lines of the generated program to the code lines of the sample program, and the second one is to evaluate the ratio of average execution time of the generated program to the execution time of the sample program. Reducing the percentage of the program lines r_l is shown in equation (11). The l_o and l_g represent the number of code lines of the sample program and the average number of code lines of the generated programs, respectively. The reduction in program execution time percentage r_t is shown in equation (12). The t_o and t_g stand for the execution time of the sample program and the average execution time of the generated programs, respectively.

$$r_l = \left(1 - \frac{l_g}{l_o}\right) \times 100\%, \quad (11)$$

$$r_t = \left(1 - \frac{t_g}{t_o}\right) \times 100\%, \quad (12)$$

3.6. Predetermining the Number of Generated Programs Statistically. This section is to explore how many of the generated programs produced by GPT-2 can guarantee at least a few predetermined programs existing and having the pass ratio of code similarity checking with sample program over 90%. Therefore, we first take a count of code the generated programs that have been generated from a single

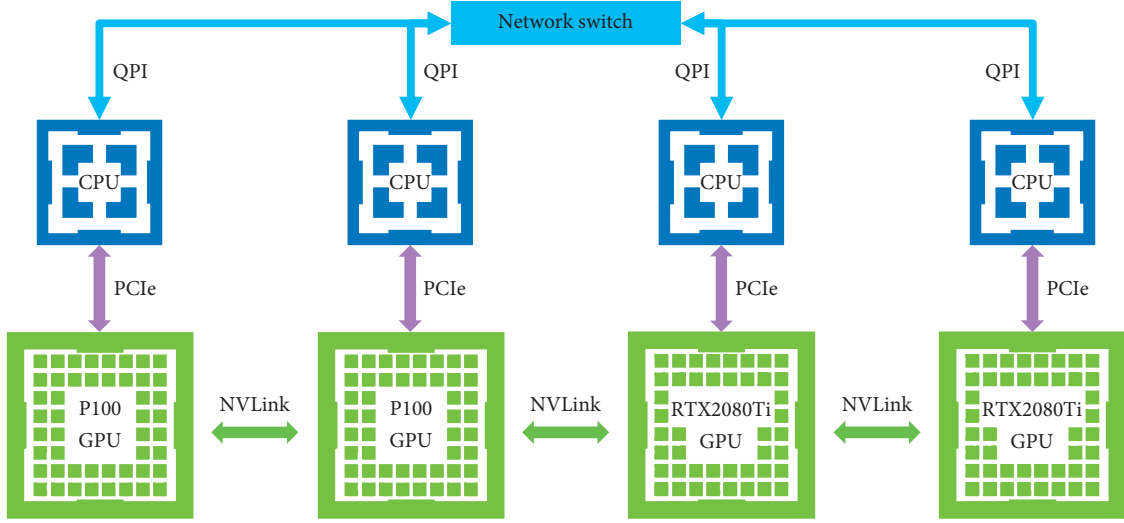


FIGURE 12: GPU cluster workstation.

TABLE 3: Keyword search confusion matrix.

	Relevant, positive class	Nonrelevant, negative class
Retrieved	True positives (TP)	False positives (FP)
Not retrieved	False negatives (FN)	True negatives (TN)

sample program, having the pass ratios of code similarity checking over 90%, and then make it possible to calculate the percentage q_i as shown in equation (13). Among them, N_{si} is the total number of generated programs produced from a single sample program in which x_{si} is the number of generated programs whose pass ratio of similarity checking is more than 90%. After every percentage q_i mentioned above has been obtained, equation (14) represents the average percentage of q_i where t stands for the total number of sample programs in a single example sentence. Then, we make a judgment to determine how many misjudgments are there in x_{si} , and then calculate the percentage of misjudgments q_{mi} , as shown in equation (15), where x_{msi} is the number of misjudgments within the generated programs having a pass ratio of code similarity checking over 90%. Then, the average percentage of misjudgments q_m can be obtained as shown in equation (16).

$$q_i = \frac{x_{si}}{N_{si}}, \quad i = 1, 2, \dots, t, \quad (13)$$

$$q = \frac{\sum_{i=1}^t q_i}{t}, \quad (14)$$

$$q_{mi} = \frac{x_{msi}}{x_{si}}, \quad i = 1, 2, \dots, t, \quad (15)$$

$$q_m = \frac{\sum_{i=1}^t q_{mi}}{t}. \quad (16)$$

Next, we count how many of the generated programs y_{si} , which are generated from a single sample program, have pass

ratios of below 90%, and then make it possible to calculate the percentage u_i , as shown in equation (17). After all of the existing percentages mentioned above have been calculated, equation (18) can give an average percentage u of the generated programs having the pass ratio of code similarity checking less than 90%. Then, we make a judgment to determine how many misjudgments are there in y_{si} and then calculate the percentage of misjudgments u_{mi} , as shown in equation (19), where y_{msi} is the number of misjudgments within the generated programs having the pass ratio of code similarity checking below 90%. Then, the average percentage of misjudgments u_m can be obtained as shown in equation (20).

$$u_i = \frac{y_{si}}{N_{si}}, \quad i = 1, 2, \dots, t, \quad (17)$$

$$u = \frac{\sum_{i=1}^t u_i}{t}, \quad (18)$$

$$u_{mi} = \frac{y_{msi}}{y_{si}}, \quad i = 1, 2, \dots, t, \quad (19)$$

$$u_m = \frac{\sum_{i=1}^t u_{mi}}{t}. \quad (20)$$

Then, we add up the number of programs generated from all the sample programs to get as shown in equation (21). After obtaining N_g , q , q_m , u , and u_m through the above calculations, equation (22) can calculate an average probability P_{gq} [35] of the generated programs having the pass ratio over 90%, and they are generated from the sample programs. We assume that there are j programs with a pass ratio of code more than 90%, so $P(K_j|P_{gt})$ represents the probability of the pass ratio of similarity checking more than 90% for these j programs is real, as shown in equation (23) [36], where $P(K_j \cap P_{gt})$ means the probability of there are at most j programs having the pass ratio of code similarity

checking over 90% in the generated programs, and K_j indicates there are j programs having the pass ratio of code similarity checking over 90% within the generated programs produced by the code transform model at a time. According to the abovementioned statistics, we know that the probability of j programs having the pass ratio of code similarity checking over 90% is $P(K_j|P_{gt})$. We can deduce how many programs must be generated to guarantee that there are j programs having the pass ratio of code similarity checking more than 90%, as shown in equation (24), where N is the total number of programs to be generated.

$$N_g = \sum_{i=1}^t N_{si}, \quad (21)$$

$$P_{gq} = \frac{N_g \cdot q \cdot (1 - q_m) + N_g \cdot u \cdot u_m}{N_g}, \quad (22)$$

$$P(K_j|P_{gt}) = \frac{P(K_j \cap P_{gt})}{P_{gt}}, \quad (23)$$

$$N \cdot P(K_j|P_{gt}) \geq K_j. \quad (24)$$

Let's take 4 sample programs as an example. Each sample program generates 500 programs individually and then counts how many programs have the pass ratio of code similarity checking more than 90% among the 500 programs, and then calculates the pass ratio of each generated program to have more than 90%, the average percentage can be calculated to be 3%. Then, we judge the programs whose pass ratio of code similarity checking is over 90%. After judging the number of misjudgments, we calculate the individual percentages, and finally the average percentage of misjudgments is 1%.

Moreover, we judge how many of these 500 programs have the pass ratio of code similarity checking not more than 90%, and then calculate their percentages. After calculation, we can find that the pass ratio of code similarity checking for 4 sample programs does not have an average of more than 90%. The percentage is 97%, and then we judge the programs whose pass ratio of code similarity checking is not more than 90%, judge the number of misjudgments, and then calculate the percentage. Finally, we can get the average misjudgment ratio of 2%.

Based on the above statistics, we can find that the pass ratio of code similarity checking in the generated program is really over 90% and the average probability is 4.91%. Then we want to know how many programs need to be generated to guarantee 5 programs having the pass ratio of code similarity checking over 90%. The above related values can be substituted into equation (24) to obtain the answer. As a result, at least 100 generated programs must be produced to guarantee 5 of them having the pass ratio of code similarity checking more than 90%.

4. Experimental Results and Discussion

4.1. Experimental Design. Four experiments are carried out in the following. The first experiment is to make word segmentation to select keywords and to optimize keyword retrieval. The second experiment is to search for sample programs and generate a number of preliminary programs based on the predetermined number of generated programs statistically. The third experiment is to analyze the pass ratio of code similarity checking and classify few preliminary programs as qualified programs. Finally, applying LCS conformity checking between each qualified program and sample program intents to find out the one with the highest LCS conformity, and this qualified program denotes a pocket program in the last experiment.

The experimental setting has exemplified four sample sentences into practice for all experiments. After word segmentation, the keyword retrieval optimization was implemented in two aspects. The first one is to filter the redundant keywords, and the second is to add the required keywords. Evaluation metrics, such as accuracy, recall, and F1-Score, are used to measure the performance of keyword retrieval. The sample programs associated with keywords are obtained from GitHub [37] and both of them are stored in a semantic database. In the XAMPP server, the correlation table of a semantic database consists of the several fields: keyword, sample program names, sample program paths, generated model paths, and pocket program path, as shown in Figure 13. The objective of this paper is to improve the performance of sample programs by transforming them into newly generated programs produced by GPT-2 based on two indicators: (1) to reduce the number of program code lines and (2) to decrease the program execution time.

The sample program of example 1 is related to a web crawler [38], and the corresponding keywords were "weather, traffic". The purpose of the sample program 1 was to crawl the corresponding data on the Internet to get weather forecast on that day from the Weather Center and automatically allocate the traffic congestion spots on Google Map. Next, in the sample program of example 2, the corresponding keywords are "stock, currency" that are related to exchange rate [39]. The main purpose of the sample program was to display the current currency exchange rate or the current index of the stock to the users. Third, in the sample program of example 3, the corresponding keyword is a "pets" and it's related to the web camera [40]. The objective of the web camera programming was to have the camera installed on the desktop computer at home for the video of the pet. Finally, the corresponding keywords of the sample program of example 4 are "invest, don't, insure". The system would give the user the market data analysis as per the request from investment exploration [41].

In the experimental settings, four example sentences are used to optimize keyword retrieval by filtering redundancy and adding new keywords. The example sentences are shown in Table 4.

	id	keyword	program_name	location	checkpoint
<input type="checkbox"/> edit <input type="checkbox"/> copy <input type="checkbox"/> delete	1	the weather	Web-Crawler	D:\	D:\checkpoint
<input type="checkbox"/> edit <input type="checkbox"/> copy <input type="checkbox"/> delete	2	exchange rate	Exchange-Rate	D:\	D:\checkpoint
<input type="checkbox"/> edit <input type="checkbox"/> copy <input type="checkbox"/> delete	3	photography	Web Camera	D:\	D:\checkpoint
<input type="checkbox"/> edit <input type="checkbox"/> copy <input type="checkbox"/> delete	4	investment	Market-Analysis	D:\	D:\checkpoint

FIGURE 13: Correlation table of four sample programs.

TABLE 4: Example sentences.

Example	Sentence context
Example 1	The weather is very good today, I want to know the traffic flow.
Example 2	Recently, the stock has continued to fall and is not stable, looking for currency trading or gold trading.
Example 3	How are your pets at home? I don't know what's going on in the shop.
Example 4	If you want to invest in financial management, don't insure, but deposit.

TABLE 5: NLTK word segmentation.

Example	Word segmentation
Example 1	["The", "weather", "is", "very", "good", "today", ",", "I", "want", "to", "know", "the", "traffic", "flow", "."]
Example 2	["Recently", ",", "the", "stock", "has", "continued", "to", "fall", "and", "is", "not", "stable", ",", "looking", "for", "currency", "trading", "or", "gold", "trading", "."]
Example 3	["How", "are", "your", "pets", "at", "home", "?", "I", "do", "n't", "know", "what", "s", "going", "on", "in", "the", "shop", "."]
Example 4	["If", "you", "want", "to", "invest", "in", "financial", "management", ",", "do", "n't", "insure", ",", "but", "deposit", "."]

```

In [1]: runfile('D:/eng1.py', wdir='D:')
['The', 'weather', 'is', 'very', 'good', 'today', ',', 'I', 'want', 'to', 'know', 'the', 'traffic', 'flow', '.']

['Recently', ',', 'the', 'stock', 'has', 'continued', 'to', 'fall', 'and', 'is', 'not', 'stable', ',', 'looking', 'for', 'currency', 'trading', 'or', 'gold', 'trading', '.']

['How', 'are', 'your', 'pets', 'at', 'home', '?', 'I', 'do', 'n't', 'know', 'what', 's', 'going', 'on', 'in', 'the', 'shop', '.']

['If', 'you', 'want', 'to', 'invest', 'in', 'financial', 'management', ',', 'do', 'n't', 'insure', ',', 'but', 'deposit', '.']
    
```

FIGURE 14: Screenshot of NLTK word segmentation.

4.2. Experimental Results

4.2.1. *The Experiment #1.* The above four sentences were used as the word segmentation model NLTK to select the keywords. The results of keyword selection are shown in Table 5. The screenshot is shown in Figure 14.

As mentioned above, NLTK operation that carried out word segmentation was not optimized in key words retrieval. All segmented words except for punctuation marks were selected as keywords. The selected keywords are shown in Table 6, and the screenshot is shown in Figure 15.

The initial selected keywords were consistent with the existing keywords in the semantic database as shown in Table 7. Hit keywords were “weather, traffic” in example 1, “stock, currency” in example 2, “pets” in example 3, and “invest, don't, insure” in example 4.

After selecting keywords out of the sample sentences, the accuracy of keyword-searching was on the basis of the number of hit keywords in the semantic database. Those related keywords were retrieved and denoted as true positive

(TP), and those retrieved but unrelated keywords were denoted as false positive (FP). The confusion matrix for keyword retrieval from example 1 to example 4 is shown in Tables 8–11.

After initial keyword-searching using NLTK, the first optimization method is to filter out the unrelated auxiliary words or conjunctions in the sentences. The detailed results of hit keywords in the semantic database are shown in Table 12. The related screenshot is shown in Figure 16. After the screening process, the confusion matrices after optimization for keyword retrieval from Example 1 to Example 4 are shown in Tables 13–16.

The second keyword search optimization method was to add related keywords and sample program of the semantic database. Furthermore, the number of keywords that reflected the semantic database was newly optimized. Added keywords are “today, very, good, know, flow” in Example 1, “Recently, continue, fall, gold, trading” in Example 2, “home, How, shop, know, what” in Example 3, and “want, but, deposit” in Example 4. The details of keywords in the

TABLE 6: Keywords selection out of the example sentences.

Example	Keyword
Example 1	["The", "weather", "is", "very", "good", "today", "I", "want", "to", "know", "the", "traffic", "flow"]
Example 2	["Recently", "the", "stock", "has", "continued", "to", "fall", "and", "is", "not", "stable", "looking", "for", "currency", "trading", "or", "gold", "trading"]
Example 3	["How", "are", "your", "pets", "at", "home", "I", "don", "t", "know", "what", "s", "going", "on", "in", "the", "shop"]
Example 4	["If", "you", "want", "to", "invest", "in", "financial", "management", "do", "n't", "insure", "but", "deposit"]

```
In [1]: runfile('D:/eng1.py', wdir='D:')
['The', 'weather', 'is', 'very', 'good', 'today', 'I', 'want', 'to', 'know', 'the', 'traffic', 'flow']

['Recently', 'the', 'stock', 'has', 'continued', 'to', 'fall', 'and', 'is', 'not', 'stable', 'looking', 'for', 'currency', 'trading', 'or', 'gold', 'trading']

['How', 'are', 'your', 'pets', 'at', 'home', 'I', 'don', 't', 'know', 'what', 's', 'going', 'on', 'in', 'the', 'shop']

['If', 'you', 'want', 'to', 'invest', 'in', 'financial', 'management', 'do', 'n't', 'insure', 'but', 'deposit']
```

FIGURE 15: Screenshot of selected keywords.

TABLE 7: Hit keyword in semantic database.

Example	Hit keyword in semantic database
Example 1	Weather, traffic
Example 2	Stock, currency
Example 3	Pets
Example 4	Invest, don't, insure

TABLE 12: The result of filter keywords in the semantic database in Experiment 1.

Example	Hit keyword in semantic database
Example 1	Weather, traffic
Example 2	Stock, currency
Example 3	Pets
Example 4	Invest, don't, insure

TABLE 8: Example 1: Confusion matrix for keyword retrieval in Experiment 1.

	Related	Unrelated
Retrieved	2	6
Not retrieved	0	0

```
In [1]: runfile('D:/eng1.py', wdir='D:')
['weather', 'traffic']
['stock', 'currency']
['pets']
['invest', 'do', 'n't', 'insure']
```

FIGURE 16: The screenshot of keywords retrieval with filter optimization.

TABLE 9: Example 2: Confusion matrix for keyword retrieval in Experiment 1.

	Related	Unrelated
Retrieved	2	7
Not retrieved	0	0

TABLE 13: Example 1: keyword retrieval confusion matrix with filter optimization in Experiment 1.

	Related	Unrelated
Retrieved	2	1
Not retrieved	0	0

TABLE 10: Example 3: Confusion matrix for keyword retrieval in Experiment 1.

	Related	Unrelated
Retrieved	1	7
Not retrieved	0	0

TABLE 14: Example 2: keyword retrieval confusion matrix with filter optimization in Experiment 1.

	Related	Unrelated
Retrieved	2	1
Not retrieved	0	0

TABLE 11: Example 4: Confusion matrix for keyword retrieval in Experiment 1.

	Related	Unrelated
Retrieved	3	4
Not retrieved	0	0

TABLE 15: Example 3: keyword retrieval confusion matrix with filter optimization in Experiment 1.

	Related	Unrelated
Retrieved	1	0
Not retrieved	0	0

TABLE 16: Example 4: keyword retrieval confusion matrix with filter optimization in Experiment 1.

	Related	Unrelated
Retrieved	3	4
Not retrieved	0	0

TABLE 17: The result of added keywords in the semantic database in Experiment 1.

Example	Hit keyword in semantic database
Example 1	today, weather, very, good, know, traffic, flow
Example 2	recently, continue, stock, fall, currency, gold, trading
Example 3	home, pets, how, shop, know
Example 4	want, invest, don't, insure, but, deposit

TABLE 18: Experiment 1: Example 1: added optimized keyword search confusion matrix.

	Related	Unrelated
Retrieved	7	1
Not retrieved	0	0

TABLE 19: Experiment 1: Example 2: added optimized keyword search confusion matrix.

	Related	Unrelated
Retrieved	7	2
Not retrieved	0	0

TABLE 20: Experiment 1: Example 3: Added optimized keyword search confusion matrix.

	Related	Unrelated
Retrieved	6	2
Not retrieved	0	0

TABLE 21: Experiment 1: Example 4: added optimized keyword search confusion matrix.

	Related	Unrelated
Retrieved	6	1
Not retrieved	0	0

TABLE 22: Evaluation indexes of Example 1 in Experiment 1.

Evaluation index (%)	Initial selection (%)	Filter optimization (%)	Add optimization (%)	Average (%)
Accuracy	25	67	75	94
Recall	100	100	100	100
F1-score	40	80	86	97

TABLE 23: Evaluation indexes of Example 2 in Experiment 1.

Evaluation index (%)	Initial selection (%)	Filter optimization (%)	Add optimization (%)	Average (%)
Accuracy	22	67	78	89
Recall	100	100	100	100
F1-score	36	80	88	94

TABLE 24: Evaluation indexes of example 3 in Experiment 1.

Evaluation index (%)	Initial selection (%)	Filter optimization (%)	Add optimization (%)	Average (%)
Accuracy	13	100	75	88
Recall	100	100	100	100
F1-score	22	100	86	93

TABLE 25: Evaluation indexes of Example 4 in Experiment 1.

Evaluation index	Initial selection (%)	Filter optimization (%)	Add optimization (%)	Average (%)
Accuracy	30	75	86	89
Recall	100	100	100	100
F1-score	46	86	92	94

semantic database are shown in Table 17. The confusion matrices for the keyword search from Example 1 to Example 4 are shown in Tables 18–21.

The Evaluation Metrics such as the accuracy, recall, and F1-Score of the initial keywords retrieval, filter keywords retrieval, and newly added keywords retrieval were evaluated. The results are shown in Tables 22–25. Since all keywords in the segmented words in sample sentences were selected by default, the recall was 100%.

4.2.2. The Experiment #2. The second experiment was based on four sample programs obtained from GitHub. Corresponding to the keywords found in the experiment #1, the corresponding keywords were extracted from the natural language sentences and will be applied to the sample programs. The correspondence of the sample programs and keywords are listed in Table 26.

In order to transform sample programs to the high-performance generated programs, a code transform model GPT-2 generated 100 preliminary programs, and its time consuming was also recorded at the same time. In this experiment, a total of five rounds was performed, and the estimated average time to generate a program in real-time was summarized in Table 27.

4.2.3. The Experiment #3. In experiment #3, the comparison of Simhash similarity checking between the above four sample programs and the programs generated by GPT-2 were performed on a cluster GPU workstation. The aim of this experiment was to find out how many completed programs would have similarity percentage greater than or equal to the default pass ratio set by the user earlier. In this experiment, the diagram of qualified ratio distribution set the X-axis as the similarity percentage, ranging from 0% to 100% with 20% as the separation interval. The Y-axis was set as the number of programs in the percentage ratio interval. For each corresponding sample program, code transform model GPT-2 will generate 100 programs denoted as

TABLE 26: The list of example programs in Experiment 2.

Example	Sample program	Keyword
Example 1	Web crawler	Weather, traffic
Example 2	Exchange rate	Stock, currency
Example 3	Web camera	Pets
Example 4	Market-analysis	Invest, don't, insure

TABLE 27: Estimated time to generate one hundred programs (unit: second).

Sample program	First round	Second round	Third round	Fourth round	Fifth round	Average
Sample program 1	170	185	163	183	147	170
Sample program 2	142	188	176	150	178	167
Sample program 3	171	144	168	196	183	172
Sample program 4	162	186	175	144	152	164

```

File Edit Format View Help
from bs4 import BeautifulSoup
import requests

winning_Numbers_Sort_lotto = ['Lotto649Control_history_d1Query_No1_', 'Lotto649Control_hist

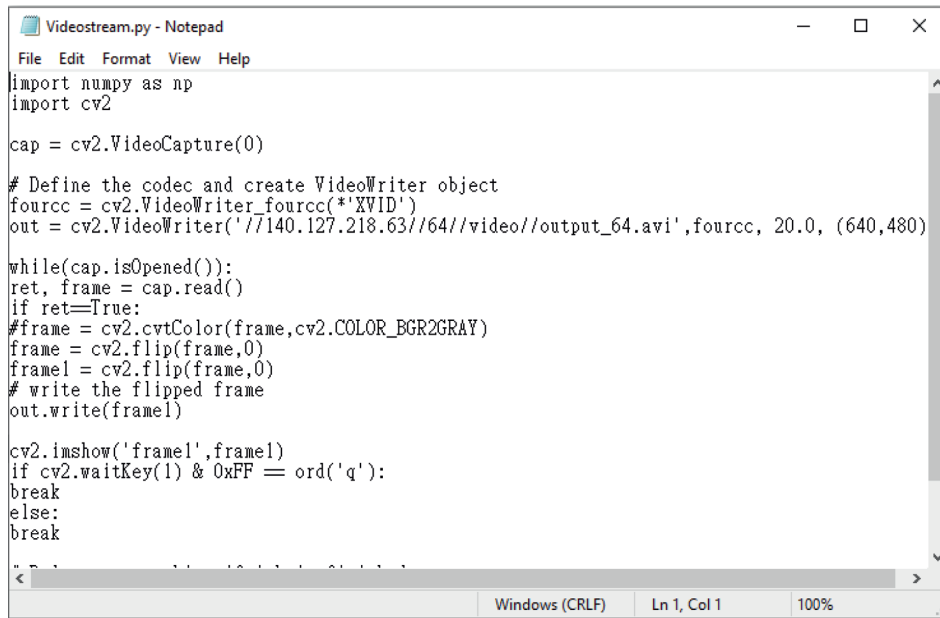
def search_winning_numbers(css_class):
    global winning_Numbers_Sort_lotto
    if(css_class != None):
        for i in range(len(winning_Numbers_Sort_lotto)):
            if winning_Numbers_Sort_lotto [i] in css_class:
                return css_class
    def parse_tw_lotto_html(data_Info,number_count):
        data_Info_List = []
        data_Info_Dict = {}
        tmp_index = 0
        for index in range(len(data_Info)):
            if (index == 0):
                data_Info_List.append(data_Info[index].text)
            else:
                if(index % number_count != 0):
                    data_Info_List.append(data_Info[index].text)
                else:
                    data_Info_Dict[str(tmp_index)] = list(data_Info_List)
                    data_Info_List= []
    
```

FIGURE 17: Sampled preliminary program associated with sample program 1 in Experiment 3.

```

File Edit Format View Help
import pandas
dfs = pandas.read_html( "http://rate.bot.com.tw/xrt?Lang=zh-TW")
currency = dfs[0]
currency = currency.ix[:,0:5]
currency.columns = [u' type' ,u' buy' ,u' sell' ,u' buy' ,u' sell' ]
currency[u' type' ] = currency[u' type' ].str.extract( '\\((\\w+)\\)' )
currency.to_excel( 'currency.xlsx' )
import pandas
dfs = pandas.read_html( "http://rate.bot.com.tw/xrt?Lang=zh-TW")
currency = dfs[0]
currency = currency.ix[:,0:5]
currency.columns = [u' type' ,u' buy' ,u' sell' ,u' buy' ,u' sell' ]
currency[u' type' ] = currency[u' type' ].str.extract( '\\((\\w+)\\)' )
currency.to_excel( 'currency.xlsx' )
import pandas
dfs = pandas.read_html( "http://rate.bot.com.tw/xrt?Lang=zh-TW")
currency = dfs[0]
currency = currency.ix[:,0:5]
currency.columns = [u' type' ,u' buy' ,u' sell' ,u' buy' ,u' sell' ]
currency[u' type' ] = currency[u' type' ].str.extract( '\\((\\w+)\\)' )
currency.to_excel( 'currency.xlsx' )
import pandas
dfs = pandas.read_html( "http://rate.bot.com.tw/xrt?Lang=zh-TW")
currency = dfs[0]
    
```

FIGURE 18: Sampled preliminary program associated with sample program 2 in Experiment 3.



```

Videostream.py - Notepad
File Edit Format View Help
import numpy as np
import cv2

cap = cv2.VideoCapture(0)

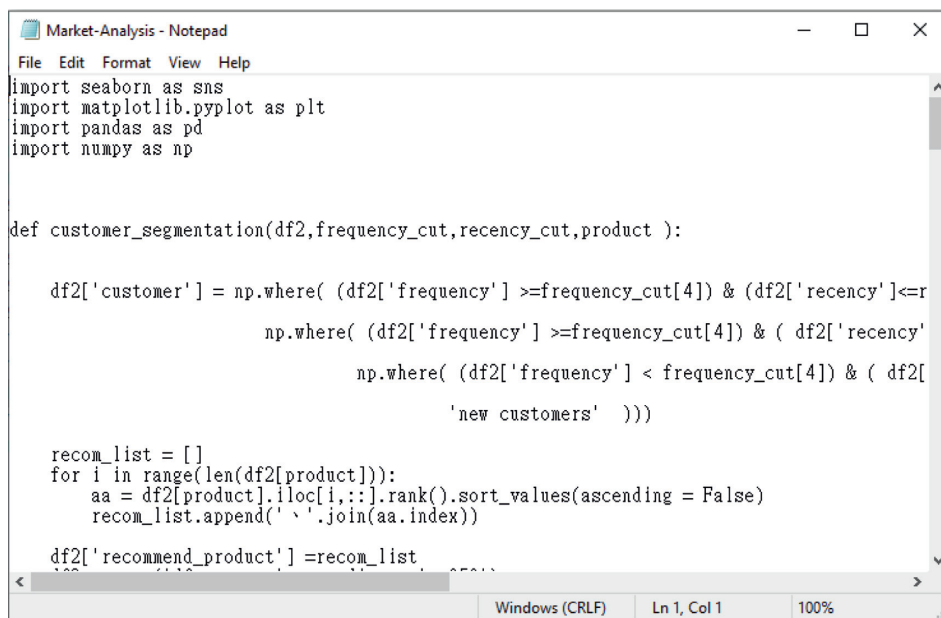
# Define the codec and create VideoWriter object
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('//140.127.218.63//64//video//output_64.avi',fourcc, 20.0, (640,480))

while(cap.isOpened()):
    ret, frame = cap.read()
    if ret==True:
        #frame = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
        frame = cv2.flip(frame,0)
        frame1 = cv2.flip(frame,0)
        # write the flipped frame
        out.write(frame1)

    cv2.imshow('frame1',frame1)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
    else:
        break

```

FIGURE 19: Sampled preliminary program associated with sample program 3 in Experiment 3.



```

Market-Analysis - Notepad
File Edit Format View Help
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

def customer_segmentation(df2,frequency_cut,recency_cut,product ):

    df2['customer'] = np.where( (df2['frequency'] >=frequency_cut[4]) & (df2['recency'] <=r
        np.where( (df2['frequency'] >=frequency_cut[4]) & ( df2['recency'
            np.where( (df2['frequency'] < frequency_cut[4]) & ( df2[
                'new customers' )))

    recom_list = []
    for i in range(len(df2[product])):
        aa = df2[product].iloc[i,:].rank().sort_values(ascending = False)
        recom_list.append('`'.join(aa.index))

    df2['recommend_product'] =recom_list

```

FIGURE 20: Sampled preliminary program associated with sample program 4 in Experiment 3.

preliminary programs. Samples of the preliminary programs are shown in Figures 17–20. The pass ratios of the preliminary programs are shown in Figures 21–24. We define the pass ratio as how many programs out of 100 preliminary programs would have the similarity falling within the range of 80%–100%. In other words, the pass ratios of this experiment associated with sample programs 1, 2, 3, and 4 were 40%, 30%, 38%, and 37%, respectively, based on 100 generated preliminary programs, and those preliminary programs are referred to as qualified programs instead.

4.2.4. The Experiment #4. According to 4 examples demonstrated in the experimental setting, the fourth experiment first attempts to verify whether the execution result of the qualified program meets a certain proportion of conformity with the sample program. After qualified programs have been compiled successfully, the qualified programs are executed individually, and then the execution result of each qualified program is compared with the execution result of a corresponding sample program using LCS conformity. Their execution results are shown in Figures 25–28. The program execution result is converted into ASCII code through LCS algorithm to compare

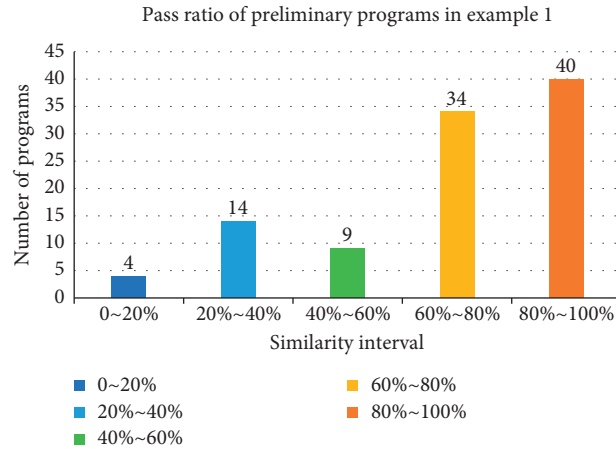


FIGURE 21: The pass ratio of the preliminary programs associated with sample program 1 in Experiment 3.

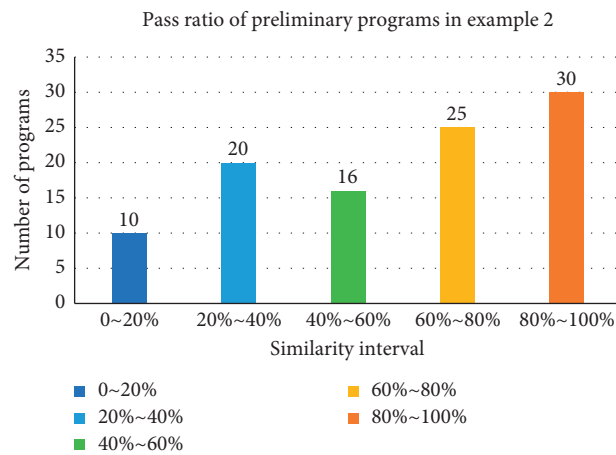


FIGURE 22: The pass ratio of the preliminary programs associated with sample program 2 in Experiment 3.

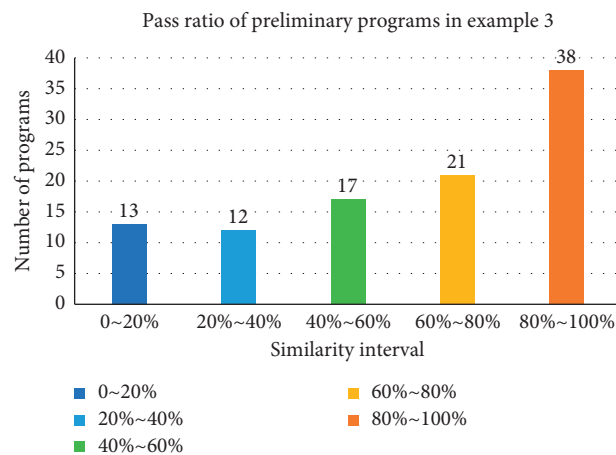


FIGURE 23: The pass ratio of the preliminary programs associated with sample program 3 in Experiment 3.

the conformance. As a result, the qualified program with the highest pass ratio of LCS conformity is called the best qualified program and is also designated as a pocket program. The experimental result is listed as shown in Table 28.

Next, we have compared the performance of the above four best qualified programs produced by GPT-2 with their corresponding sample programs. The evaluation includes (1) to compare the number of code lines between the sample

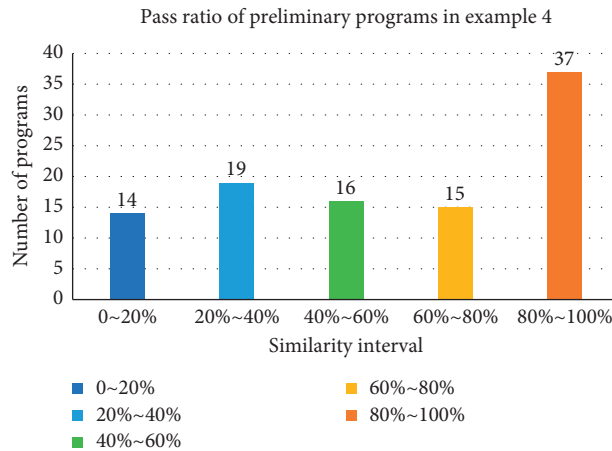


FIGURE 24: The pass ratio of the preliminary programs associated with sample program 4 in Experiment 3.

```
In [1]: runfile('C:/Users/user/Desktop/Web-Crawler.py', wdir='C:/Users/user/Desktop')
{'0': ['16', '17', '18', '19', '33', '45', '25'], '1': ['08']}
```

(a)

```
In [1]: runfile('C:/Users/user/Desktop/untitled9.py', wdir='C:/Users/user/Desktop')
{'0': ['16', '17', '18', '19', '30', '41', '25'], '1': ['08']}
```

(b)

FIGURE 25: Program execution result in example 1. (a) Execution result of sample program 1. (b) Execution result of the best qualified program.

```
In [1]: runfile('C:/Users/user/Desktop/Market-Analysis.py', wdir='C:/Users/user/Desktop')
Recommend you to invest in Sinopec Yuanta 02 purchase 01
opening 8.4,increase 0.6%
```

(a)

```
In [1]: runfile('C:/Users/user/Desktop/untitled16.py', wdir='C:/Users/user/Desktop')
Recommend you to invest in Sinopec Yuanta 02 purchase 01
open 8.7 ,increase 0.66%
```

(b)

FIGURE 26: Program execution result in example 2, (a) Execution result of sample program 2. (b) Execution result of the best qualified program.

```
In [1]: runfile('C:/Users/user/Desktop/Videostream.py', wdir='C:/Users/user/Desktop')
Turn on the mini camera
Watch the situation in the store
```

(a)

```
In [1]: runfile('C:/Users/user/Desktop/untitled19.py', wdir='C:/Users/user/Desktop')
Turn on the mini camera
look the situation in the store
```

(b)

FIGURE 27: Program execution result in example 3, (a) Execution result of sample program 3. (b) Execution result of the best qualified program.

program and the corresponding best qualified program and (2) to compare the execution time of the sample program and the corresponding best qualified program. To understand how much improvement was made in the speed of program execution, the average number of code lines of 100

preliminary programs and the average execution time for the best qualified programs were carefully examined. The estimation results are listed in Tables 29 and 30, respectively.

Regarding the credibility and validity of the findings in this study, the comparison of performance evaluation

```
In [1]: runfile('C:/Users/user/Desktop/Exchange-Rate.py', wdir='C:/Users/user/Desktop')
type:USD
Buy 29.1925 dollars
Sell 29.6030 dollars
```

(a)

```
In [1]: runfile('C:/Users/user/Desktop/untitled22.py', wdir='C:/Users/user/Desktop')
type : USD
Buy 29.8964 dollars
Sell 29.1079 dollar
```

(b)

FIGURE 28: Program execution result in example 4, (a) Execution result of sample program 4. (b) Execution result of the best qualified program.

TABLE 28: Comparison of program execution results based on LCS conformity (unit: %).

Number of ASCII code	Example 1	Example 2	Example 3	Example 4
Number of ASCII code counted from the sample program execution output	62	89	57	51
Number of ASCII code counted from the best qualified program execution output	62	87	57	51
Number of ASCII code counted from LCS output	61	86	56	49
LCS conformity	98.38%	97.72%	98.24%	96.07%
Average LCS conformity	97.60%			

TABLE 29: Number of code lines comparison.

Number of code lines or percentage	Example 1	Example 2	Example 3	Example 4
Number of source code lines of the sample program	291	175	138	153
Average of number of code lines of 100 preliminary programs	174	117	99	108
Reduced the percentage of code lines	40.34%	32.96%	28.02%	29.54%
Average percentage reduction	32.71%			

TABLE 30: Program execution time comparison (unit: second).

Time or percentage	Example 1	Example 2	Example 3	Example 4
Sample program execution time	8.35	11.83	9.74	11.86
Best qualified program execution time	6.97	8.05	7.07	9.32
Reduced the percentage of program execution time	16.59%	31.93%	27.40%	21.43%
Average percentage reduction	24.34%			

among source-code to source-code transform models is listed in Table 31. The performance evaluation includes (1) average time of generating a single instruction (s), (2) reduced the percentage of code lines (%), and (3) conformity of program execution results (%). As a result, the proposed approach with GPT-2 outperforms Java-Codetool and CodeGeneration. Java-Codetool and CodeGeneration introduced in the paper [16] are of Java code transform models. However, this article didn't present the execution results of generated programs. Thus, there is no information about the conformity of program execution results for Java-Codetool and CodeGeneration models in Table 31.

5. Discussion

In the first experiment, after NLTK word segmentation, keyword-searching optimization has shown that if there were fewer hit keywords, filtering operation would require screening more irrelevant keywords to improve the

F1-Score. In contrast, the alternative was to add keywords to the semantic database and improving the accuracy and F1-Score of keywords retrieval significantly. In the second experiment, the number of generated program was produced through GPT-2 based on predetermined the number of generated programs statistically and an average of 100 programs were generated for every corresponding sample program and on average, each program was generated for about 1 second. The number of code lines for generating 100 programs was reduced by an average of 32.71%, so that the code review could save about 30% of time due to less number of code lines. After the code similarity checking for the generated preliminary programs was completed in the third experiment, the fewer programs having the highest pass ratio were selected as the qualified programs in this part. After the qualified programs were compiled successfully, the last experiment is the LCS conformity checking between the qualified programs and the sample program where the ratio of conformity in fact exceeds 97.60% on average and

TABLE 31: Performance evaluation of code generation model.

Time or percentage	Java-codetool	CodeGeneration	Proposed approach with GPT-2
Average time of generating a single instruction (s)	0.17	0.15	0.01
Reduced the percentage of code lines (%)	0%	0%	24.34%
Conformity of program execution results (%)	—	—	97.60%

The symbol “—” stands for the information that is not available so far.

the one with the highest ratio of conformity is chosen as a pocket program. Regarding the performance of the qualified programs, the average execution time of the generated program was reduced by 24.34%.

The experiments have shown that the system could not only quickly generate programs but also greatly improve the program execution efficiency. Based on transfer learning and few-shot learning, GPT-2 has implemented the best level of code transform task to produce newly generated programs for significantly improving its execution efficiency. Furthermore, with the powerful cloud platform, Hadoop or Spark, people in the future can realize collective learning to integrate all of the small data sets provided from different units, evolve into sophisticated machine learning applications, create a large enough semantic database, and achieve a great computing power.

6. Conclusion

This study proposes a novel transform method to replace the existing programs inside the voice assistant machine with high-performance generated programs through GPT-2 reasonably. In particular, this paper introduces theoretical estimation in statistics to infer at least a number of generated programs needed so as to guarantee the best one can be founded within them. As a result, in terms of performance evaluation, the average number of newly generated code lines decreased by 32.71%, and the average execution time of the program decreased by 24.34%. This proved that the system could not only quickly generate programs but also greatly improve the performance of program execution. In the future, we are looking for developing a new method to speed up data retrieval in the semantic database and finding the revision of Simhash and Longest Common Subsequence (LCS) to achieve better accuracy of measuring the code similarity and the conformity of program execution results.

Data Availability

The Sample Program.rar data used to support the findings of this study have been deposited in the <https://drive.google.com/file/d/1KYDeoO9s8kA94U9-CW1AsdB0ZZNr4Hcy/view?usp=sharing> repository. The sample sentence data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

B.R.C. and P.-W.S. conceived and designed the experiments; H.-F.T. collected the experimental dataset and proofread the paper; and B.R.C. wrote the paper.

Acknowledgments

This work was fully supported by the Ministry of Science and Technology, Taiwan, Republic of China, under grant numbers MOST 105-2221-E-390-013-MY3 and MOST 109-2622-E-390-002-CC3.

References

- [1] F. Nasirian, M. Ahmadian, and O. K. D. Lee, “Ai-based voice assistant systems: evaluating from the interaction and trust perspectives,” in *Proceedings of the Twenty-third Americas Conference on Information Systems*, pp. 1–10, Boston, MA, USA, 2017.
- [2] H. Shah, K. Warwick, J. Vallverdú, and D. Wu, “Can machines talk? Comparison of Eliza with modern dialogue systems,” *Computers in Human Behavior*, vol. 58, pp. 278–295, 2016.
- [3] S. Arora, V. A. Athavale, H. Maggu, and A. Agarwal, “Artificial intelligence and virtual assistant—working model,” *Mobile Radio Communications*, vol. 140, pp. 163–171, 2020.
- [4] K. Ethayarajh, “How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings,” 2019, <http://arxiv.org/abs/1909.00512>.
- [5] NLTK, <https://github.com/nltk/nltk>.
- [6] GPT-2: 1.5B Release, <https://openai.com/blog/gpt-2-1-5b-release/>.
- [7] D. M. J. Lazer, M. A. Baum, Y. Benkler et al., “The science of fake news,” *Science*, vol. 359, no. 1360, pp. 1094–1096, 2018.
- [8] W. Wagner, S. Bird, E. Klein, and E. Loper, “Natural language processing with Python, analyzing text with the natural language toolkit,” *Language Resources and Evaluation*, vol. 44, no. 4, pp. 421–424, 2010.
- [9] J. Park, M.-S. Chen, and P. S. Yu, “Using a hash-based method with transaction trimming for mining association rules,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 9, pp. 813–825, 1997.
- [10] Longest common subsequence problem, https://en.wikipedia.org/wiki/Longest_common_subsequence_problem.
- [11] H. Chung, J. Park, and S. Lee, “Digital forensic approaches for Amazon Alexa ecosystem,” *Digital Investigation*, vol. 22, pp. S15–S25, 2017.
- [12] E. Reiter and R. Dale, “Building applied natural language generation systems,” *Natural Language Engineering*, vol. 3, no. 1, pp. 57–87, 1997.
- [13] M. Geittle and R. Olsson, “Using automatic programming of design improved variants of differential evolution,” in *Proceedings of the 2017 21st Asia Pacific symposium on intelligent*

- and evolutionary systems(IES), Hanoi, Vietnam, December 2017.
- [14] Á. Beszédés, R. Ferenc, T. Gyimóthy, A. Dolenc, and K. Karsisto, "Survey of code-size reduction methods," *ACM Computing Surveys (CSUR)*, vol. 35, no. 3, pp. 223–267, 2003.
 - [15] L. Li, J. Yang, Z. Liu, and L. Bao, "The research and application of web page code automatic generation technology," in *Proceedings of the IEEE 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce*, pp. 5246–5249, Zhengzhou, China, August 2011.
 - [16] Z. Li, Y. Jiang, X. J. Zhang, and H. Y. Xu, "The metric for automatic code generation," *Procedia Computer Science*, vol. 166, pp. 279–286, 2020.
 - [17] R. Collobert, J. Weston, B. Leon, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "natural language processing (almost) from scratch," *Journal of Machine Learning Research*, 2011.
 - [18] K. Gimpel, N. Schneider, B. O'Connor et al., "Part-of-speech tagging for twitter: annotation, features, and experiments," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Shortpapers*, pp. 42–47, Portland, OR, USA, 2011.
 - [19] Y. Cui, S. Ahmad, and J. Hawkins, "Continuous online sequence learning with an unsupervised neural network model," *Neural Computation*, vol. 28, pp. 2474–2504, 2016.
 - [20] B. Myagmar, J. Li, and S. Kimura, "Cross-domain sentiment classification with bidirectional contextualized transformer language models," *IEEE Access*, vol. 7, pp. 163219–163230, 2019.
 - [21] S. Schoenmackers, J. Davis, O. Etzioni, and D. Weld, "Learning first-order horn clauses from web text," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, vol. 10, pp. 1088–1098, EMNLP, Brussels, Belgium, 2010.
 - [22] E. Gilbert, "Widespread underprovision on reddit," in *Proceedings of the 2013 Conference of Computer Support Cooperative Work (CSCW'13)*, pp. 803–808, San Antonio, TX, USA, 2013.
 - [23] GPT-2 size, <https://kknnews.cc/zh-tw/tech/5rlolbk.html>.
 - [24] C. Sadowski and G. Levin, "Simhash: hash-based similarity detection," 2007, <https://www.webrankinfo.com/dossiers/wp-content/uploads/simhash.pdf>.
 - [25] K. Anil, "Jain and jian-jiang feng, "latent fingerprint matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, pp. 88–100, 2010.
 - [26] L. Zhang, Y. Zhang, J. Tang, Ke Lu, and T. Qi, "Binary code ranking with weighted hamming distance," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1586–1593, Portland, OR, USA, 2013.
 - [27] Hamming distance, http://www.code10.info/index.php%3Foption%3Dcom_content%26view%3Darticle%26id%3D59:hamming-distance%26catid%3D38:cat_coding_algorithms_data-similarity%26Itemid%3D57.
 - [28] J. Tiedemann, "Automatic construction of weighted string similarity measures," in *Proceedings of Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pp. 213–219, 1999.
 - [29] D. D. Dvorski, *Installing, Configuring, and Developing with XAMPP*, "Skills Canada, Skills, Waterloo, Canada, 2007.
 - [30] M. Bouache, J. L. Glover, and J. Boukhobza, "Analysis of memory performance: mixed rank performance across microarchitectures," in *Proceedings of the International Conference on High Performance Computing*, pp. 579–590, Innsbruck, Austria, 2016.
 - [31] D. Foley and J. Danskin, "Ultra-performance pascal GPU and NVLink interconnect," *IEEE Micro*, vol. 37, pp. 7–17, 2017.
 - [32] M. Abdullahi, M. A. Ngadi, and S. i. M. Abdulhamid, "Symbiotic organism search optimization based task scheduling in cloud computing environment," *Future Generation Computer Systems*, vol. 56, pp. 640–650, 2016.
 - [33] H. Huang, H. Xu, X. Wang, and W. Silamu, "Maximum F1-score discriminative training criterion for automatic mispronunciation detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, pp. 787–797, March 06, 2015.
 - [34] X. Deng, Q. Liu, Y. Deng, and S. Mahadevan, "An improved method to construct basic probability assignment based on the confusion matrix for classification problem," *Information Sciences*, vol. 340, pp. 250–261, 2016.
 - [35] D. E. Over, C. Hadjichristidis, J. St. B. T. Evans, S. J. Handley, and S. A. Sloman, "The probability of causal conditionals," *Cognitive Psychology*, vol. 54, no. 1, pp. 62–97, 2007.
 - [36] T. Flaminio, L. Godo, and H. Hosni, "Boolean algebras of conditionals, probability and logic," *Artificial Intelligence*, vol. 286, Article ID 103347, 2020.
 - [37] E. Kalliamvakou, G. Gousios, B. Kelly, L. Singer, and M. Daniel, "German, and daniela damian, "the promises and perils of mining GitHub," MSR," in *Proceedings of the 11th Working Conference on Mining Software Repositories 2014*, pp. 92–101, Hyderabad, India, 2014.
 - [38] Web-crawler, <https://github.com/jwlin/web-crawler-tutorial>.
 - [39] Exchange-rate, <https://github.com/wert30678/Repo/blob/master/exchange-rate>.
 - [40] Web camera, <https://github.com/chinaev/python-video-streaming>.
 - [41] Market-analysis, <https://github.com/HowardNTUST/Marketing-Data-Science-Application/blob/master/Python-RFM-RF-basics/>.

Research Article

Key Performance Indicators for the Integration of the Service-Oriented Architecture and Scrum Process Model for IOT

Mengze Zheng ¹, **Islam Zada**,² **Sara Shahzad**,² **Javed Iqbal**,² **Muhammad Shafiq**,³
Muhammad Zeeshan,⁴ and **Amjad Ali**²

¹College of Digital Technology and Engineering, Ningbo University of Finance and Economics, Ningbo, Zhejiang 315175, China

²Department of Computer Science, University of Peshawar, Peshawar, Pakistan

³Cyberspace Institute of Advance Technology, Guangzhou University, Guangzhou, China

⁴Kohat University of Science and Technology, Kohat, Pakistan

Correspondence should be addressed to Mengze Zheng; zhengmengze@nbufe.edu.cn

Received 10 November 2020; Revised 28 December 2020; Accepted 7 January 2021; Published 2 February 2021

Academic Editor: Muhammad Arif Shah

Copyright © 2021 Mengze Zheng et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

An important aspect in any business process lifecycle is management of the performance, where performance requirements on business processes are specified as Key Performance Indicators (KPIs) with target values which are to be achieved in a certain analysis period. A KPI is a business metric used to measure and evaluate the individual capability, maturity, complexity, and agility of a business process in the development environment. This study designed four general KPIs for the integration of SOA and scrum to bring further advancement in these approaches for IIoT. The study also identified some common metrics which will give help to software developers and, especially, to those who want to apply SOA and scrum integration. These metrics will play a critical role of bridging the strategy and concepts of improvements with operational activities. The identified KPIs will help to measure the business agility, quality and value, team efficiency, and complexity of scrum- and SOA based projects. Software development organizations can also practice these KPIs to know where to focus their resources to deliver the ultimate business profit. So, software business organizations could better align their business projects and IT investments with the rapid market change and deliveries.

1. Introduction

In the present era of dynamic business environment, flexibility to welcome change and adapting to it efficiently and cost effectively is pertinent to the success of any business organization. Flexibility and change adoption are key attributes of Service-Oriented Architecture (SOA) and agile software development processes [1]. Although the notion of agility is quite visible on both sides, still the integration of the two diverse concepts (architectural frame work and development process) should be well thought of before employing them for a software development project [2]. Therefore, use of an appropriate agile process for the SOA-based application development to adopt major requirements, modifications, and changes even during application building along with the conservation of software superiority and quality is

essential [3]. Scrum is one of the agile software developments techniques, through which a system is developed efficiently and rapidly by means of regular, frequent, and complete releases permitting the participants and stakeholders in the project to get their hands on the application in order to review and test it through a retrospective meeting. A prototype concept or model can be developed into a useful progressive and creative system by means of an iterative and incremental process whereby feedback is given by the stakeholders, based upon the rapid successive releases of the software.

The scrum method openly addresses design and goes up with disparaging such as big design up print (such as the water fall approach) to depress this attitude. While most Service-Oriented Architecture (SOA) teams are almost predominantly serviceable players gathered around circles of

services. The SOA's nature inspires specific team makeup and styles of communication in the interior teams which are the dominion of policies such as scrum practices. We can say that scrum is like the human hands that work in the mitt. While SOA is like the mitt, where the scope is enterprise wide, the scrum process is about the mode you can build the application part that is supported by software. Up to our study effort, most of the scrum and SOA principles are not in conflict with each other. Applications development thorough scrum without a clear and strong idea of the aims of the organization will be useless. SOA without a strong image of exactly how to design and build it genuinely using scrum process model rules is a waste of resource and time.

Estimation and measurement of system development cost and revenue impact, as well as other scrum and SOA metrics, is vital to any prominent business organization. Measuring the value and tracking changes to the metrics are critical as your system's services progress grows and its range increases. A confirmed manner to prove an SOA and scrum's industry value is through their respective KPIs. We can say that metrics are the language of KPIs, KPIs using it where your business associates understand it. These KPIs can give the means to measure the agility, complexity, efficiency, and value of the scrum and SOA team for those who want to use the scrum and SOA combination. We have identified sixteen different metrics for SOA and scrum which are discussed in detail in the coming section of this paper. Some of these are most important to keep on track the business value for the cross combination of scrum and SOA. In light of the using the SOA and scrum combination, we have combined the individually identified metrics into four common metrics, which make the KPIs for cross combination of SOA and scrum. The team velocity, business agility, product quality, and effort review become the Key Performance Indicators (KPIs) for the scrum and SOA development approach (SSDA).

The first part of this paper generally explains the SOA and scrum as an agile process model, and the secondary source for this study is the existence work of different scholars which is discussed as a literature review in the second part. The identified scrum, SOA metrics, and the integration of these two approaches are discussed in the third and fourth sections. The identified KPIs are presented in the fifth section. The sixth part of this study describes the analysis and discussion, while conclusions, implications, and limitations and future work are presented in sections seven, eight, and nine, respectively.

2. Literature Review

Researchers and professionals have a mixed opinion about the estimation and measurement, similarity, and compatibility of the scrum and SOA approaches. Critics emphasize differences among SOA and agile approaches, arguing that SOA and agile are standing at different development directions: SOA is an architecture and agile is a methodology [4], SOA working in a top-down manner and agile as inherently a bottom-up approach [5]. SOA is an architectural framework and follows a set of principles, whereas agile is a

process model and more at the practice level. Some of the researchers also claim that SOA-based systems are developed and deployed differently from traditional developments [1]. Also, there are many challenges such as stockholders involvement, business and IT alignment, and reuse of assets. To overcome such type of problems, agility and service orientation are better integrated. It is notable that scrum and SOA share similar concerns, such as responsiveness to changes, new ways of working, flexibility, and business understanding [6]. Different authors discussed these two terminologies and their in titrations in different ways, which are discussed individually in the following subsections as:

2.1. Service-Oriented Architecture (SOA). SOA is an architectural framework and approach to design, develop, manage, and deploy a software application and software infrastructure in which all applications are structured into business logic called services that are network executable and accessible. In other words, SOA agrees to the integration of applications, users, and existing system into a flexible architecture that can easily accommodate changes when it is needed in a system [7]. SOA is regarded as one of the best approaches for distributed application development.

SOA allows reusing the functionality of existing systems, rather than building again from scratch. This feature of reusability in the SOA-based applications maximizes economic benefits for the organization [8]. Each service in SOA performs autonomously but is not isolated from the whole. Each service encapsulates a specific logic in the problem domain. The main features of SOA are reusability, loose coupling, service contract, autonomy, abstraction, discoverability, and statelessness [9]. SOA agrees to the integration of applications, users, and existing system into a flexible architecture that can easily accommodate changes when it is needed in a system [10]. SOA is regarded as one of the best approaches for distributed application development [11].

2.2. Scrum as Agile Software Development. An agile process model tends to focus on iterations and client suggestions to improve performance and allow for the predictability of varying requirements. Agile software development (ASD) is the development process through which a system is developed efficiently and rapidly by means of regular, frequent, and complete releases permitting the participants and stakeholders in the project to get their hands on the application in order to review and test it through an agile retrospective meeting. A prototype concept or model can be developed into a useful progressive and creative system by means of an iterative and incremental process whereby feedback is given by the stakeholders, based upon the rapid successive releases of the software. Scrum is an agile methodology which is the most standard way of introducing agility due to its flexibility and straight forwardness [12] and a popular management agile method in industry. Agile development of applications in an enterprise surrounding can be challenging because of the compound nature team members and their environments [13]. The agile software

development process facilitates discovery of better ways of developing software by promoting individual and teamwork [14]. Agile processes are planned to maintain early and fast development of software application. This is made possible by dividing the development process into sprints (or iterations) where sprint stresses on the delivery of working product that provides value to both the project and customer [15, 16]. Scrum, as the most commonly used agile process, highlights empirical feedback, team self-management, and struggling to build properly tested product increments within short iterations [17].

2.3. Integration of Scrum and SOA and KPIs. Flexibility and change adoption are key attributes for Service-Oriented Architecture (SOA) and agile software development processes. Although the notion of agility is quite visible on both sides, still the integration of the two diverse concepts (architectural frame work and development process) should be well thought of before employing them for a software development project. Therefore, the use of an appropriate agile process for the SOA-based application development, to adopt major requirements modifications and changes even during application building along with the conservation of software superiority and quality, is essential [18]. SOA and scrum are both the development approaches but following different directions. In the services development scenario, the SOA approach follows the bottom-top approach while scrum follows the top-bottom approach as using a process development methodology. Here, some questions arise such as how these different approaches can be compatible and measurable with each other when applying on the same task simultaneously? Is SOA also following the same measurement approach like the scrum process? If the answer is yes, then why not to use the same measurement approach for both scrum and SOA? How these two approaches could be integrated with each other?

Today, estimation and measurement of system development cost and revenue impact, as well as other scrum and SOA metrics, is vital to any prominent business organization. Measuring the value and tracking changes to the metrics are critical as your system's services progress grows and its range increases. A sure way to validate software development business value is through measurements metric which will make the Key Performance indicators (KPIs). KPI uses some type of terminologies or language that can be understood by your business colleagues, which are metrics. They can provide you the resources and knowledge to measure an SOA and scrum-related project to real business enlargements.

Here, we are using metrics to estimate software development progress in getting a continuing vision and short-range quarterly objectives. These metrics can make KPIs which will be the leading and guidance force that could synchronize goals with daily operating performance. Different authors have discussed different metrics for SOA and scrum individually using different terminologies, but among those metrics, we have discussed the most important metrics which can give more benefits to market and those people

applying the combination of SOA and scrum in a software development project. We have discussed the most important scrum and SOA metrics which are summarized in this section.

2.4. Research Problem and Research Contribution. Although, SOA allows reusing the functionality of existing systems, rather than building again from scratch. This feature of reusability in SOA-based applications maximizes economic benefits for the organizations [3]. Each service in SOA performs autonomously but is not isolated from the whole. Each service encapsulates a specific logic in the problem domain. The other features of SOA are loose coupling, service contract, autonomy, abstraction, discoverability, and statelessness, while scrum is an agile methodology which is a standard way of introducing agility due to its flexibility and straight forwardness [4]. It is a popular agile management method in industry. Agile development of applications in an enterprise surrounding can be challenging because of the compound nature of team members and their environment [5]. The scrum process facilitates discovering better ways of developing software by promoting individuals, as well as teams [6].

Although SOA and agile approaches are generally viewed with related concerns, still there is no clear definition of organization and setup of both approaches in a single environment. Very little information is provided as to what will be the impact of this integrated implementation on the important factors such as productivity, quality, agility, and innovativeness. Understanding of key performance indicators of the maturity of scrum and SOA integration is also an issue. Therefore, the proposed study is aimed at analyzing the compatibility of scrum and SOA with rules and practices for scrum and SOA integrated application. This can be carried out by defining KPIs of the integrated scrum and SOA environment using which an organization can go ahead with a successful management of the SOA project using the scrum process model. SOA and scrum are two different approaches that follow different directions. In the services development scenario, the SOA approach follows the top-down approach (services are built on the top of the SOA system) while scrum follows the bottom-up approach (starting from initial planning to prototype delivery) as using a process development methodology [19, 20]. Question arises that how these different approaches are compatible with each other when employed together for a development process? Another question is whether SOA also follows the agility just like scrum process. If the answer is yes, then how? Finally, how these two approaches could be integrated with each other to get benefits offered by both individually?

In this study, an SOA based application development project is selected as a case study, for which the scrum process model is used as a development methodology. This SOA-based industrial project is named as M4S (Mineral resource, Mapping, Modeling, and Management System). The project development and deployment perspective includes eight core modules that constitute the overall project framework. Large modules are subdivided into smaller

modules for better organization and management. The system is developed following standard phases of the scrum development approach. The researcher has participated in this project to analyze and evaluate the development processes. As already discussed that scrum and SOA are working on different directions, the researcher has analyzed the compatibility, diversity, and similarity of these different approaches. Scrum and SOA metrics are analyzed to measure their flexibility, complexity, agility, and team efficiency. After the analysis of scrum and SOA metrics, four general KPIs are designed for the measurement of these different approaches.

The identified KPIs are team velocity, business agility, quality assurance, and effort review. These KPIs will help software business organizations to understand where to commit resources in order to deliver the optimal business value. It will also help to align software development business projects and IT investments with respect to market change. These KPIs will guide practitioners to measure and improve their integrated scrum and SOA approach.

3. KPIs for Scrum and SOA Integration

An important aspect in the business process lifecycle is estimation, measurement, and management of the performance of business processes. Performance requirements on business processes are specified as Key Performance Indicators (KPIs) with target values which are to be achieved in a certain analysis period [21]. The KPI is a business metric which measures the individual capability, maturity, complexity, and agility of a business process in development environment. To bring further advancement in both SOA and agile approaches, we have identified some common metrics which might give help to software developers and, especially, to those who want to apply SOA and agile combination, as metrics plays the critical role of bridging the strategy and concepts of improvements with operational activities [22, 23]. It encapsulates the process, people, tools, and techniques that result in seamless reporting and the governance of the metrics to the required stakeholders including the executive leadership that eventually owns and directs Continual Service Improvement in an organization. Metrics is concerned with the process, procedures, tools, and templates that integrate to provide the benefits to the organization.

The main purpose of using metrics for the software development process is

- (i) To align business objectives with IT
- (ii) To help achieve compliance requirements for business operations
- (iii) To drive operational excellence of IT services

SOA and scrum metrics are designed to measure and evaluate the complexity, agility, effort estimation, and flexibility of an organization's business solution [64]. These metrics are grouped into two major categories: scrum metric and agile SOA metrics, which are depicted in Figures 1 and 2.

3.1. Scrum Metrics. The flexibility and ability to quickly respond to market fluctuations makes agile development methods attractive for companies operating in a market-driven context, despite the fact that the long-term impact of adopting these principles and their applicability in the market-driven context are, to a large extent, unknown. Existing studies and experience reports from application of agile methods are mostly isolated to evaluating the performance of these methods on software development activities, such as increasing the developer's efficiency and producing better quality code. For this purpose, this study identified and analyzed the eight scrum metrics given in Table 1, which will keep a scrum team on track, where X_1 , X_2 , X_3 , X_4 , X_5 , X_6 , X_7 , and X_8 represent the completed stories vs. committed stories, team velocity, quality delivered to the customer, team enthusiasm, proper and improper use of scrum practices, retrospective process improvement, team communication, and reduction in the project and maintenance expenses eighth metric, respectively.

3.2. SOA Metrics. In the selection of applicable metrics and estimated KPIs to define the level of realization of business organization goals, during deployment, system factors need to be perfected to boost SOA KPIs. Also, the system managing arrangement is set up to bring together dimensions to support demarcated metrics, monitoring, service-level agreement parameters, and runtime reworking. The purpose of SOA metrics is to measure and evaluate the complexity, agility, effort estimation, and flexibility of the SOA and agile solution system. SOA measurements are taken to acquire the maximum consideration and openly relate to successful SOA implementations in any development organization. There are a small number of measurement regions that should be looked into by any group and could be used as a starting point. This study also identified the eight SOA metrics which are given in Table 2, where Y_1 , Y_2 , Y_3 , Y_4 , Y_5 , Y_6 , Y_7 , and Y_8 represent the revenue per service, development time for a service, service quality assurance, new created and used as a percentage of total, service accessibility and usability, average time to service development, number of services reused, and violations of architecture policies, respectively.

These are the measures that appear to acquire the maximum consideration and openly relate to fruitful SOA employments and implementation using scrum as the development process model.

4. Integration of Scrum and SOA

As already discussed, SOA and scrum are two different approaches that follow different directions. In the services development scenario, the SOA approach follows the top-down approach (services are building in the top of SOA system) while scrum follows the bottom-up approach (starting from initial planning to prototype delivery) as using a process development methodology [19, 20]. Question arises that how these different approaches are compatible

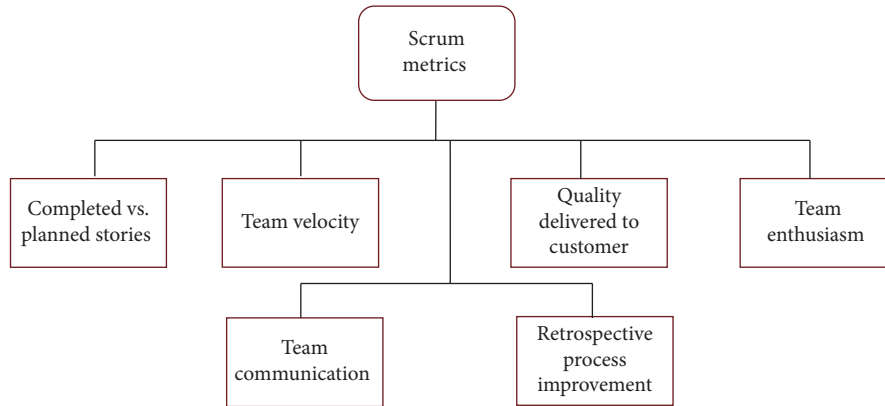


FIGURE 1: Scrum metrics.

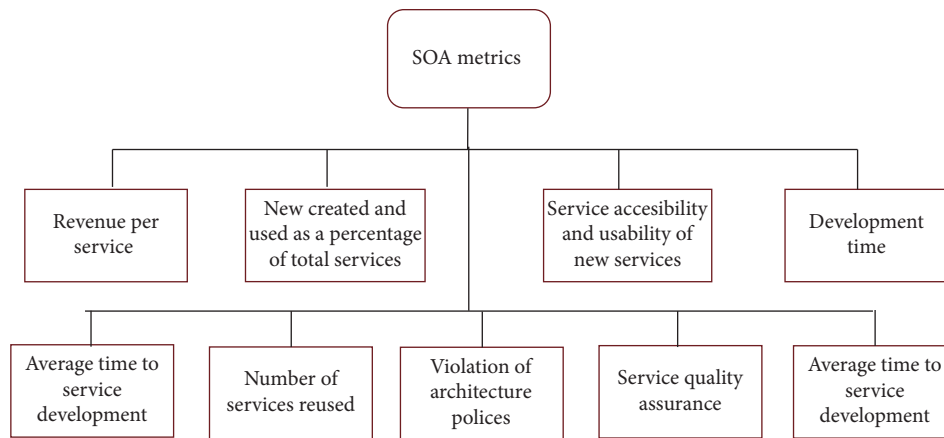


FIGURE 2: SOA metrics.

with each other when employed together for a development process? Another question is whether SOA also follows the agility just like the scrum process. If the answer is yes, then how? Finally, how these two approaches could be integrated with each other to get benefits offered by both individually? This section refers to these questions and also discusses SOA and scrum metrics having commonalities. This study identified different metrics; among them, some are most important which can provide more value to business and to those whose aim is to use SOA and scrum together in a software development venture. Some of the scrum and SOA metrics are used for common purpose sharing common features. Table 3 shows the scrum and SOA metrics which share some of the common features and goals.

5. Key Performance Indicators of Scrum and SOA

Measuring of revenue and other process, product, and project metrics is essential for the development and improvement of software development organizations [24]. Measuring scrum and SOA individually and tracking changes to these metrics are very difficult but critical for business process success and improvement [25]. The success

and improvement of scrum and SOA integration can be analyzed through their respective KPIs, which are designed from their individual metrics. KPIs translate the business performance in terms where the business associates understand. These KPIs provide a way to measure the agility, complexity, efficiency, and value of scrum and SOA teams [26]. This study has identified different metrics for scrum and SOA (discussed in detail in the previous section). The individual metrics of scrum and SOA which share features are mapped into common metrics to provide KPIs for scrum and SOA integration. The four resulting KPIs are meant to keep the business value on track for the cross combination of scrum and SOA. Therefore, Team Velocity (TV), Business Agility (BA), Product Quality (PQ), and Effort Review (ER) are the key performance indicators for the integrated scrum and SOA approach (ISSA), which are the scrum development process applied to develop SOA, based software application. The summary of these KPIs is given in Table 4.

Details about each KPI are provided in the following sections.

5.1. *Team Velocity (TV)*. Table 4 identifies team velocity as the first KPI which is evaluated from the combined scrum and SOA metrics, namely, team velocity, completed stories

TABLE 1: Summary of the scrum.

Scrum metrics	Description
Completed stories vs. committed stories (X_1)	This metric is capable of identifying the team capabilities to compare and measure the committed stories planned per sprint and actual progress of scrum team that how many stories are completed by the team per sprint
Team velocity (X_2)	The velocity metric measures the consistency of the team's estimates from sprint to sprint. The measure is made by comparing story points completed in this sprint with points completed in the previous sprint
Quality delivered to the customer (X_3)	This metric is related to the efficiency and skills of developers, customer needs, and project requirements. In this metric, we measure whether the product is built according to the customer needs or not and does every sprint provide value to the customer and become a potentially releasable piece of the product?
Team enthusiasm (X_4)	The enthusiasm measurement is carried out by observing various sprint meetings or simply asking team members the following questions included in a questionnaire which was distributed among the team members: Do you feel happy? If not, then why? How motivated do you feel?
Proper and improper use of scrum practices (X_5)	This metric measures whether team members follow the scrum rules and manifes properly or not
Retrospective process improvement (X_6)	This type of metrics measures the team's capability to revise, within the scrum process improvement, context, and practices, its development process to make it more effective for the coming sprint
Team communication (X_7)	This metrics is an individual measure of how are the product owner, scrum master, customers, and other team members directed straightforward and exposed to communications. The scrum master's responsibility was to observe and listen to the team members and product owner, and other stockholders will get indications as how everyone is well collaborating and communicating throughout the sprint
Reduction in the project and maintenance expenses (X_8)	This metric will be used to compute the intact project's expenses avoidance amount and just sum the total expenses for all the SOA services being leveraged. Also, to predict the total potential expenses evading at any stage and point of stage, we multiply the number of times each SOA service is planned to be leveraged by its designed expenses and sum these total

TABLE 2: Summary of the SOA.

SOA metrics	Description
Revenue per service (Y_1)	This metric is concerned with the release of a sprint; it is the imperative measure for a business organization as income for a service. This quantity takes up together the charge an organization provides and the output/efficiency accomplished based on the charge generated by the service base
Development time for a service = (Y_2)	This type of metrics measures the time required for delivering a new application, process, or service. As each scrum story card was designed in such a way that it consists of only one service/ piece of work and each story was assigned 10 to 16 hours to be completed, some services were finished before the specified time and some were finished after the specified time
Service quality assurance (Y_3)	In software engineering, the complexity is generally measuring through cycloramic complexity, and the cycloramic complexity of software is the single measurement that will regulate if your system's service is maintainable and testable. Many authors claimed that services with cycloramic complexity greater than fifty are not testable and often result in ten to twenty percent more maintenance effort than those services whose cycloramic complexity is less than ten
New created and used as a percentage of total (Y_4)	Organizations with poor SOA domination and governance usually see out-of-control service growth, by means of high ratio of new services as a percentage of total services. Uncontrolled software development scrum teams often look to build new service after new service, not thinking about redesigning the existing established services in order to achieve the desired value. Not only does this drive increase the services total cost but also decrease the average revenue per service, showing low development production
Service accessibility and usability (Y_5)	This metric defines the time in percent at which percent the services are used by the end users. It is the measurement of the complete build and delivered services system, from the bugs-free services to functional data centers. Less than 99.9% needs to be carried on instantly since they bear customer agreement
Average time to service development (Y_6)	Average time to service development provides a statistical calculation for measurement of services with some certainty of the mean time to arise an individual service

TABLE 2: Continued.

SOA metrics	Description
Number of services reused (Y_7)	Here, the reusability of service means reusing of existing services when they are needed in other parts of the SOA system because many services are accruing in multiple places which are already build for that purpose. Reusability of services is the main property of the SOA system, through which we can get the agility and can remove the complexity, as well as the service development time, cost, and resources
Violations of architecture policies (Y_8)	This measure defines the violations of architecture policies during the development process and its frequencies that how much rules and regulations are violated. The development process will have control points that will check adherence to the specific architecture policies that have been selected

TABLE 3: Scrum and SOA metrics which share the common features and goals.

S.N	Scrum	SOA	Commonalities
1	Completed stories vs. planned stories	New created and used as a percentage of total services	The main purpose of these two metrics is to measure the ratio and percentage of completed work (services) developed in one sprint
2	Team velocity	Development time and average development time to develop a service	These three metrics measure the team progress in terms of per sprint and time required for a service which is to be completed in a particular sprint within time. So, these metrics could be combined into a team velocity metric which will be considered as a metric for SOA and scrum integration
3	Quality delivered to the customer	Service quality assurance	The aim of this metric is to measure the service quality when applying the scrum development process model. The quality is a common feature for these both metrics which can be combined to make a metric for SOA and scrum integrations measurements
4	Team enthusiasm	4. Violation of architecture policies	When the team members are happy, satisfied, and working in a comfortable environment, then they will communicate with each other friendly and collaboratively. They will have full attention and focus on product development, and through this, product quality will remain standard. Also, they will willingly follow the preplanned architecture policies and rules. We can also see that when the scrum team is happy and in a restful environment, they can develop large number of services of high quality in a small amount of time. So, we can say that the “team enthusiasm and communication” metrics of scrum and “violations of architecture policies” and “average time to service development” metrics are dependent on each other; these can have an effect on project when these are not concentrated. These metrics are used to measure to measure the behavior that how they follow rules and policies in the development environment
5	Team communication	5. Average time to service development	
6	Retrospective process improvement	Service accessibility and usability	These two metrics can be integrated together to represent a common metric for both scrum and SOA because the retrospective meeting is held at the last of all practices of scrum in which the overall activities could be revived. When the services are developed in a sprint, a review meeting will be arranged in which we can test the developed service functionality and usability that how to access the service and how it is working
7	Technical debt management	Reduction in the project and maintenance expense	The main purpose of these two metrics is to reduce the product development cost through best management and utilization of resources and team members’ skills. These two metrics can be integrated in one common metric for the combined use of scrum and SOA approaches

versus committed stories, new services created and used as percentage of total services, and services development time. All of these metrics measure output performance of teams or, in other words, production of a team, which is a common

goal of the scrum and SOA approach. Therefore, these metrics are combined in the form of an integrated KPI for the scrum and SOA approach. Figure 3 and equation (1) show participating factors of team velocity.

TABLE 4: Integrated KPIs for scrum and SOA.

S. NO	Name of the KPI	Common metrics of scrum and SOA
1	Team velocity (TV)	(i) Team velocity (ii) Completed stories vs. committed stories (iii) New services created and used, as the percentage of total services (iv) Services development time
2	Business agility (BA)	(i) Development time of a service (ii) Team velocity (iii) Team enthusiasm (iv) Revenue per sprint
3	Product quality (PQ)	(i) Quality delivered (ii) Service quality assurance (iii) Service accessibility and usage (iv) Revenue per service
4	Effort review (ER)	(i) Retrospective meeting (ii) Violation of architecture policies (iii) Team communication (iv) Enthusiasm

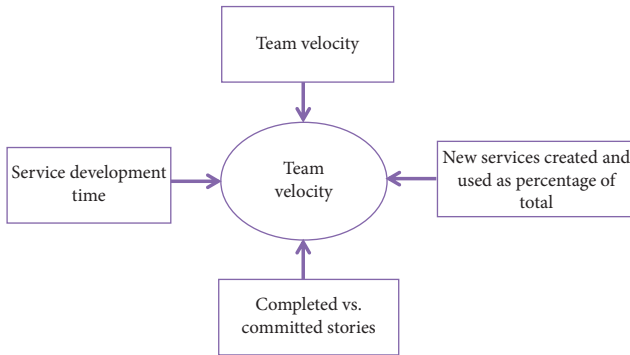


FIGURE 3: KPI No.3. Participating factor of SOA and scrum in team velocity.

$$TV = X_1 + X_2 + Y_2 + Y_4. \quad (1)$$

5.2. *Business Agility.* Figure 4 and equation (2) show the participating factors of SOA and scrum in “Business Agility.”

$$BA = X_1 + X_7 + Y_1 + Y_2. \quad (2)$$

Business agility is the second KPI which is designed from the combined of scrum and SOA metrics named as development time of a service, team velocity, team enthusiasm, and revenue per sprint. This KPI shows rapid development of SOA services and scrum “team velocity.” It shows the numbers of story points completed by the scrum team in a particular sprint. Scrum and SOA agility depend upon the progress of the scrum team and service development. Business agility and participating metrics are shown in Figure 4.

5.3. *Product Quality.*

$$PQ = X_1 + Y_1 + Y_3 + Y_5. \quad (3)$$

The four metrics of scrum and SOA, namely, quality delivered to the customer, service quality assurance, service

accessibility and usage, and revenue per sprint indirectly measure the software product quality. Therefore, these metrics are mapped to define product quality KPI, which will represent the service quality delivered by the scrum team in a specific sprint within time and budget. The mapped metrics to present the product quality KPI are shown in Figure 5.

5.4. *Effort Review.* Figure 6 and equation (4) show the participating factors of SOA and scrum in “Effort Review.”

$$ER = X_4 + X_6 + X_7 + Y_8. \quad (4)$$

Effort review is the fourth KPI designed by combining scrum and SOA metrics such as retrospective meeting, violation of architecture policies, team communication, and enthusiasm metrics. This KPI will present the review of a sprint’s daily scrum process which examines the performance in previous sprints with respect to people, relationships, process tools, and violation of architecture policies during the development of SOA services. It will also present the scrum team’s behavior and communication that they followed in the development process. This KPI along with its mapped metrics is depicted in Figure 6.

6. Analysis and Discussion

Software measurement and estimation is important for any process improvement initiative. Software metrics allow qualities of interest to be measured and evaluated in order to identify potential problems. Metrics provide insight into the costs and benefits of a potential solution. Unfortunately, existing individual scrum and SOA metrics are comparatively immature, and there is no generic measurement and metrics available for the integrated application of scrum and SOA for a software development project [27]. There is also a misunderstanding in identifying desired measurable properties of both scrum and SOA in the context of

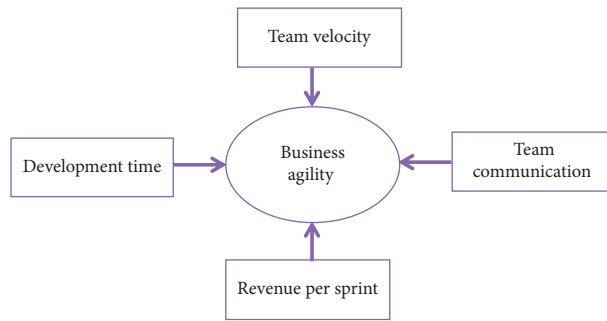


FIGURE 4: Participating factors of SOA and scrum in “Business Agility.”

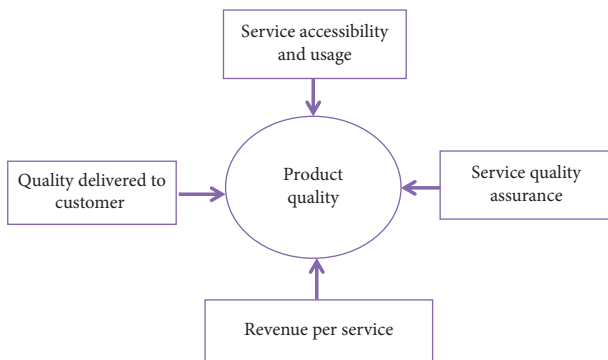


FIGURE 5: Participating factors of SOA and scrum in “Product Quality.”

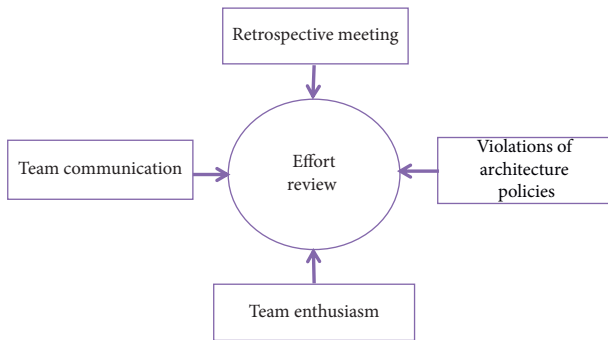


FIGURE 6: Participating factors of SOA and scrum in “Effort Review.”

integration. Another problem for identifying metrics for integrated use of scrum and SOA is that both focus on respective properties, factors, and tools and then collection of data for individual specific factors measured in the integrated environment. Hence, individual factors do not work in this context [28].

The KPIs identified in this study solve these problems and provide a basis to measure business agility, quality value, team efficiency, and complexity of scrum- and SOA-based projects. These KPIs will help to understand where to commit resources in order to deliver the optimal business value. It will also help to align software development business projects and IT investments with respect to market

change. These KPIs will guide practitioners to measure and improve their integrated scrum and SOA approach.

As some of the authors have discussed different metrics for SOA and scrum individually in different papers, among them, we have identified the most important metrics which can give more benefits to market and those people applying the combination of SOA and scrum in a software development project. Through these metrics, the development process can be streamlined when a proper measure is taken. Among these scrum and SOA metrics, some of metrics present common goals which can provide more agility, flexibility, and compatibility.

The scrum process model and service-oriented architecture (SOA) approaches buoy up business people to be liable for the significance and value provided by different software development experts and effort. These metrics will support endorsed organizations and businesses that can make insolent software investments and scrum teams that provide software services with highest value repeatedly and rapidly. One benefit of value-driven work is to deliver valuable features to customers quickly in the development may come to be self-funding during the progress of the plan [29, 30]. “Business Value Delivered” is that what the scrum team would be optimizing and enhancing for and what the business organization will be following as a key performance indicator [31]. Software services value can be best determined by the interested parties and scrum team together, at the level of specific software development groups that can be given a specific cost and customer value measure [32].

In this study, we have identified the most critical and important metrics because measuring too many metrics may not lead to project success. These metrics will give valuable information by not only minimal effort but also correct information that helps the development team to progress in their learning and reaching the objectives. Some common properties or features are found between scrum metrics and SOA metrics are the general organization of a metric, but variances in the methods to measure, for example, measuring the scrum team velocity and SOA services developed per unit time. Some of the scrum metrics use individual measurements for computing metrics such as progress of a member. Basing user stories on story points that are estimated by a team, it cannot be compared between different teams because the developed features may not have the same behavior and functions. Some common measures should be

taken to connect the scrum and SOA metrics, e.g., the scrum team velocity metric and number of services developed per unit time. As described earlier, agile has the focus on people and should, therefore, include this factor in future measurements. The main purpose of Service-Oriented Architecture (SOA) is to make the whole enterprise agile by using services as the building blocks for software applications. Also, software development through the scrum process model means to increase organization agility by bring together scrum practices that could increase communication, collaboration, and feedback. Which is accurate and better? We have identified different metrics for SOA and scrum which are discussed in detail in previous sections. Some of these are most important to keep on track the business value for the cross combination of scrum and SOA. In light of the using the SOA and scrum combination, we have combined the individually identified metrics into four common metrics, which make the KPIs for the cross combination of SOA and scrum. The team velocity, business agility, product quality, and effort review become the Key Performance Indicators (KPIs) for the scrum and SOA development approach (SSDA).

Many practitioners successfully applied agile processes, especially scrum for enterprise system development. SOA is also a favored architecture for enterprise system development in many cases; therefore, the proposed integration approach will definitely come with improved results. The KPIs presented will help in identifying and also facilitating the improvement of the benefits.

7. Conclusions

Scrum and SOA are about agility that can be applied to a number of rules and principles that do not crash each other. Scrum is about delivering rapid and SOA is about architectural configuration. This way, they maintain each other in balance. One does not make sense without the other. A confirmed manner to prove SOA and scrum's industry value is through their respective KPIs. We can say that metrics are the language of KPIs. KPIs used it in the way the business associates understand it. These KPIs give a means to measure the agility, complexity, efficiency, and value of the scrum and SOA team for those who want to use the scrum and SOA in an integrated environment; the identified metrics for SOA and scrum, discussed in detail in the previous sections, are important to keep on track the business value for the integration of scrum and SOA. In light of using the SOA and scrum integration, the individually identified metrics are combined into four common metrics, which make the KPIs for integrated scrum and SOA. "Team velocity," "Business Agility," "Product Quality," and "Effort Review" are the key performance indicators (KPIs) for the scrum and SOA development approach (SSDA).

These KPIs will help to measure the business agility, quality and value, team efficiency, and complexity of scrum- and SOA-based projects. These KPIs can be practiced to know where to focus the resources to deliver the ultimate

business profit to better align your business projects and IT investments with the market change.

8. Implications of the Study

SOA and agile are a major planning and design decisions; both require proper planning and management throughout the lifecycle of the system development. The KPIs will help all stakeholders to understand imperatives of the proposed technique and will help them in planning resources, as well as major milestones for testing the performance. These metrics will play a critical role of bridging the strategy and concepts of improvements with operational activities. The identified KPIs will help to measure the business agility, quality and value, team efficiency, and complexity of scrum- and SOA-based projects. Software development organizations can also practice these KPIs to know where to focus their resources to deliver the ultimate business profit. So, software business organizations could better align their business projects and IT investments with the rapid market change and deliveries.

9. Limitations and Future Work

It will require expertise in both SOA and agile. A management overhead in the start of the agility of the approaches will enable the teams to come up with a proper application design and smooth development process. Also, we did not apply these KPI on a real case study due to the lack of funding and time barred the application of the proposed process in a real-world scenario at the time of research. We are in the process of designing a case study for validating the research contribution. Although the proposed research shows that the presented process will have many long-lasting and awaited benefits, it can further be improved by extending it for green and sustainable software development in future.

Data Availability

No data are available to support the study.

Conflicts of Interest

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

References

- [1] K. A. Abdelouhab, D. Idoughi, and C. Kolski, "Agile and user centric SOA based service design framework applied in disaster management," in *Proceedings of the 2014 1st International Conference on Information and Communication Technologies for Disaster Management (ICT-DM)*, IEEE, Algeria, North Africa, March 2014.
- [2] T. Dingsøy, N. B. Moe, T. E. Fægri, and E. A. Seim, "Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation," *Empirical Software Engineering*, vol. 23, no. 1, pp. 490–520, 2018.

- [3] F. Rago, "Self-organizing business networks, SOA and software maintenance," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, Hong Kong, China, March 2008.
- [4] J. Sedeno, G. Vázquez, M. J. Escalona, and M. Mejías, "The systematic discovery of services in early stages of agile developments: a systematic literature review," *Journal of Computer and Communications*, vol. 07, no. 7, pp. 114–134, 2019.
- [5] T. Uslander and T. Batz, "Agile service engineering in the industrial internet of things," *Future Internet*, vol. 10, no. 10, p. 100, 2018.
- [6] B. Fitzgerald and K.-J. Stol, "Continuous software engineering: a roadmap and agenda," *Journal of Systems and Software*, vol. 123, pp. 176–189, 2017.
- [7] A. Ivanyukovich, A. Yanchuk, and M. Marchese, "Towards a service-oriented development methodology," *Journal of Integrated Design and Process Science*, vol. 9, no. 3, pp. 53–62, 2005.
- [8] M. Swientek, U. Bleimann, and P. Dowland, "Service-oriented architecture: performance issues and approaches," in *Proceedings of the Seventh International Network Conference (INC 2008)*, Lulu.com, Plymouth, UK, July, 2008.
- [9] N. Joachim, *A Literature Review of Research on Service-Oriented Architectures (SOA): Characteristics, Adoption Determinants, Governance Mechanisms, and Business Impact*, AMCIS, Detroit, MI, USA, 2011.
- [10] A. N. Fajar and N. Legowo, "Services modeling based on SOA and BPM for information system flexibility improvement," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, no. 4, p. 2451, 2018.
- [11] A. Suryatmojo, E. R. Kaburuan, A. N. Fajar, S. Sutarty, and A. S. Girsang, "Financial technology integration based on service oriented architecture," in *Proceedings of the 2018 International Conference on Orange Technologies (ICOT)*, IEEE, Bali, Indonesia, October 2018.
- [12] P. Reynisdottir, *Scrum in Mechanical Product Development*, 2013.
- [13] P. Kutschera and S. Schafer, *Applying Agile Methods in Rapidly Changing Environments*, DTIC Document, Fort Belvoir, Virginia, 2004.
- [14] M. Awad, *A Comparison between Agile and Traditional Software Development Methodologies*, University of Western Australia, Perth, Australia, 2005.
- [15] W. Theunissen, A. Boake, and D. G. Kourie, "In search of the sweet spot: agile open collaborative corporate software development," in *Proceedings of the 2005 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries*, South African Institute for Computer Scientists and Information Technologists, White River, South Africa, September 2005.
- [16] D. Turk, R. France, and B. Rumpe, "Limitations of agile software processes," in *Proceedings of the Third International Conference on eXtreme Programming and Agile Processes in Software Engineering (XP 2002)*, Cambridge University Press, Alghero, Italy, May 2002.
- [17] S. Roy and M. K. Debnath, "Designing SOA based e-governance system using eXtreme Programming methodology for developing countries," in *Proceedings of the 2010 2nd International Conference in Software Technology and Engineering (ICSTE)*, IEEE, San Juan, PR, USA, October 2010.
- [18] Z. Dragičević and S. Bošnjak, "Agile architecture in the digital era: trends and practices," *Strategic Management*, vol. 24, no. 2, pp. 12–33, 2019.
- [19] J. Schiefe, G. Saurer, and A. Schatten, "Testing event-driven business processes," *Journal of Computers*, vol. 1, no. 7, pp. 69–80, 2006.
- [20] A. Fuhr et al., *Model-Driven Software Migration: Process Model, Tool Support. Migrating Legacy Applications: Challenges in Service Oriented Architecture and Cloud Computing Environments: Challenges in Service Oriented Architecture and Cloud Computing Environments*, p. 153, 2012.
- [21] A. del-Río-Ortega, M. Resinas, C. Cabanillas, and A. Ruiz-Cortés, "On the definition and design-time analysis of process performance indicators," *Information Systems*, vol. 38, no. 4, pp. 470–490, 2013.
- [22] M. Brambilla, J. Cabot, and M. Wimmer, "Model-driven software engineering in practice: second edition," *Synthesis Lectures on Software Engineering*, vol. 3, no. 1, pp. 1–207, 2017.
- [23] M. Minelli, M. Chambers, and A. Dhiraj, *Big Data, Big Analytics: Emerging Business Intelligence And Analytic Trends For Today's Businesses*, Vol. 578, John Wiley and Sons, Hoboken, NJ, USA, 2013.
- [24] H. Alahyari, T. Gorschek, and R. Berntsson Svensson, "An exploratory study of waste in software development organizations using agile or lean approaches: a multiple case study at 14 organizations," *Information and Software Technology*, vol. 105, pp. 78–94, 2019.
- [25] S. Bente, U. Bombosch, and S. Langade, *Collaborative Enterprise Architecture: Enriching EA with Lean, Agile, and Enterprise 2.0 Practices*, Morgan Kaufmann, Waltham, MA, USA, 2013.
- [26] A. W. Brown, S. Ambler, and W. Royce, "Agility at scale: economic governance, measured improvement, and disciplined delivery," in *Proceedings of the 2013 International Conference on Software Engineering*, IEEE Press, Bari, Italy, August 2013.
- [27] V. Bendinskas, "Towards mature software process," *Information Technology And Control*, vol. 34, no. 2, 2015.
- [28] T. P. Wise and R. Daniel, *Agile Readiness: Four Spheres of Lean and Agile Transformation*, Gower Publishing, Ltd, Aldershot, UK, 2015.
- [29] P. Krogdahl, G. Luef, and C. Steindl, "Service-Oriented Agility: an initial analysis for the use of Agile methods for SOA development," in *Proceedings of the 2005 IEEE International Conference on Services Computing*, IEEE, Orlando, FL, USA, July 2005.
- [30] J. Bloomberg, *The Agile Architecture Revolution: How Cloud Computing, REST-Based SOA, and Mobile Computing Are Changing Enterprise IT*, John Wiley and Sons, New Jersey, NJ, USA, 2013.
- [31] O. Ktata and G. Levesque, "Designing and Implementing a Measurement Program for Scrum Teams: what do agile developers really need and want?," in *Proceedings of the Third C* Conference on Computer Science and Software Engineering*, ACM, Montreal, Canada, May 2010.
- [32] A. Juneja, *Value Creation and Value Capture in Software Product Business: Analyzing Product Development, B2B Sales and Software Process Methodologies*, 2011.

Research Article

Analysis of Service-Oriented Architecture and Scrum Software Development Approach for IIoT

Yanqing Cui ¹, Islam Zada,² Sara Shahzad,² Shah Nazir ³, Shafi Ullah Khan,⁴
Naveed Hussain,³ and Muhammad Asshad⁵

¹College of Computer Information, Inner Mongolia Medical University, Hohhot, China

²Department of Computer Science, University of Peshawar, Peshawar, Pakistan

³Department of Computer Science, University of Swabi, Swabi, Pakistan

⁴Kohat University of Science & Technology, Kohat, Pakistan

⁵Department of IT, University of Haripur, Haripur, Pakistan

Correspondence should be addressed to Yanqing Cui; 1653446869@qq.com and Shah Nazir; snshahnzr@gmail.com

Received 7 December 2020; Revised 22 December 2020; Accepted 8 January 2021; Published 23 January 2021

Academic Editor: Sikandar Ali

Copyright © 2021 Yanqing Cui et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Flexibility and change adoption are key attributes for service-oriented architecture (SOA) and agile software development processes. Although the notion of agility is quite visible on both sides, still the integration of the two diverse concepts (architectural framework and development process) should be well thought of before employing them for a software development project. For this purpose, this study is designed to analyze the two diverse software architectural framework and development approaches, that is, SOA and Scrum process model, respectively, and their integrated environment in software project development setup perspective for Industrial Internet of Things (IIoT). This study also analyzes commonalities among Scrum process model and SOA architectural framework to identify compatibility between Scrum and SOA so that the Scrum process can be constructively used for SOA based projects. This study also examines the proper design and setup of Scrum process suitable for large-scale SOA based projects. For this purpose, an SOA based research and development project is selected as a case study using Scrum as the software development process. The project development and deployment perspective include eight core modules that constitute the overall project framework.

1. Introduction

In the present era of dynamic business environment, flexibility to welcome change and adapting to it efficiently and cost effectively are pertinent to the success of any business organization. Service-oriented architecture (SOA) is an architectural approach to design, develop, manage, and deploy a software application and infrastructure in which all the applications are structured into business services that are network accessible and executable [1]. Flexibility and change adoption are key attributes for SOA and agile software development processes. Although the notion of agility is quite visible on both sides, still the integration of the two diverse concepts (architectural framework and development process) should be well thought of before employing them

for a software development project [2]. Therefore, the use of an appropriate agile process for the SOA based application development, to adopt major requirements modifications and changes even during application building along with the conservation of software superiority and quality, is essential [3]. Agile processes (based on agile manifesto [4]) tend to focus on iterations and client suggestions to improve performance and allows for the predictability of varying requirements. Agile software development (ASD) is the development process through which a system is developed efficiently and rapidly by means of regular, frequent, and complete releases permitting the participants and stakeholders to get their hands on the application [5, 6]. The application and process are reviewed and tested through agile retrospective meetings. In this way, a prototype is

developed into useful progressive and creative system by means of an iterative and incremental process, whereby feedback is given by the stakeholders, based upon the rapid successive releases of the software.

Iteration stresses on the delivery of working software that provides grade and value to customers as well as to project. It also produces other concerned artifacts which are valuable to both customer and project. The aim of agile process is to provide workable deliverables in a dynamic way. Proponents believe that in an agile process the main concern is to provide deliverables and working products in a dynamic way without emphasizing on design models and documentation perspective [7, 8]. Scrum is one of the agile methodologies, which is a standard way of introducing agility due to its flexibility and straightforwardness [9]. It is a popular agile management method in industry. Agile development of applications in an enterprise surrounding can be challenging because of the compound nature of team members and their environment [10]. Scrum process facilitates discovering better ways of developing software by promoting individuals as well as teams [11]. Scrum divides the development process into iterations, called sprints, where a sprint stresses on the delivery of working product that provides value to both the project and customer [12]. Scrum, as the most used agile process, highlights empirical feedback, team self-management, and struggle to build properly tested product increments within short iterations [13]. The product owner, Scrum team, and Scrum master are the main three roles in Scrum. The responsibilities of the traditional project manager role are split among these three Scrum roles [14]. Scrum master is the one who maintains the process of the team, leads Scrum meetings, and makes sure that the process esteems all rules and principles of Scrum. Scrum team is a cross-functional team (having different expertise in different areas) involving developers, designers, testers, and analysts. This team is responsible for the project as a whole and is also involved in the development. Product owner is the one who represents the interest of end users and others interested in the product parties.

Scrum is based on regular meetings in which Scrum masters, product owner(s), the developers, and third parties discuss different issues concerning the development process. These meetings are the following: sprint planning meeting, daily Scrum meeting, sprint review meeting, and sprint retrospective meeting. Sprint planning meeting is held at the beginning of every sprint cycle. During this meeting, the product owner and the Scrum team define sprint goal which the sprint will attempt to achieve [15]. The success of the sprint will later be assessed during the sprint review meeting against the sprint goal. In sprint review meeting at the end of each sprint, the team demonstrates completed functionality and shows what they have accomplished during the sprint. In sprint retrospective meeting, team members review the way the team works and interacts, behavioral aspects, and improving technical skills, so that the subsequent sprint is faster, and so forth.

The Scrum process is a black box approach, where the team and processes are built and discovered dynamically during the project working, whereas nonagile approaches

such as waterfall model are signified as a white box nature of process model from the management viewpoint [14]. Planning and postmortem phases are important because these identify goals and outputs produced in a sprint. Planning is vital for overcoming any blockage in any phase or process during development. In agile terminology, the postmortem meeting known as retrospective is important for looking at the processes and creating knowledge base through best practices. Figure 1 shows the analogy of white box and black box nature of processes models.

Nonagile development methodologies are heavy-weight processes for developing software. These methodologies are based on a successive sequence of steps, for example, requirements explanation, solution structure, testing, and deployment. Traditional methodologies impose heavy documentation and define a fixed set of requirements at the start of a project. There are various traditional methodologies such as waterfall, spiral model, win-win spiral model, and unified process. The heavy-weight processes require detailed upfront plan, heavy documentation, and broad upfront design. Still traditional methodologies have their utility for large-scale projects that have predefined and considerably fixed set of requirements, although practitioners have also started to adopt agile approaches for these projects [16].

SOA is an architectural framework and approach to design, develop, manage, and deploy a software application. It is a software infrastructure in which all applications are structured into business logic called services that are network executable and accessible. In other words, SOA agrees to integration of applications, users, and existing system into a flexible architecture that can easily accommodate changes when they are needed in a system [13]. SOA is regarded as one of the best approaches for distributed application development. SOA allows reusing the functionality of existing systems rather than building again from scratch. This feature of reusability in SOA based applications maximizes economic benefits for the organizations [14]. Each service in SOA performs autonomously but is not isolated from the whole. Each service encapsulates a specific logic in the problem domain. The other features of SOA are loose coupling, service contract, autonomy, abstraction, discoverability, and statelessness [17].

Although SOA and agile approaches are generally viewed with related concerns, still there is no clear definition of organization and setup of both approaches in a single environment for IIoT. Truly little information is provided as to what will be the impact of this integrated implementation on the important factors such as productivity, quality, agility, and innovativeness.

Based on the above claims and current market needs, this study analyzes commonalities among Scrum process model and SOA architectural framework to identify compatibility between Scrum and SOA so that the Scrum process can be constructively used for SOA based projects. This study also examines appropriate design and setup of Scrum process suitable for large-scale SOA based projects. For this purpose, an SOA based research and development project is selected as a case study using Scrum as the software development

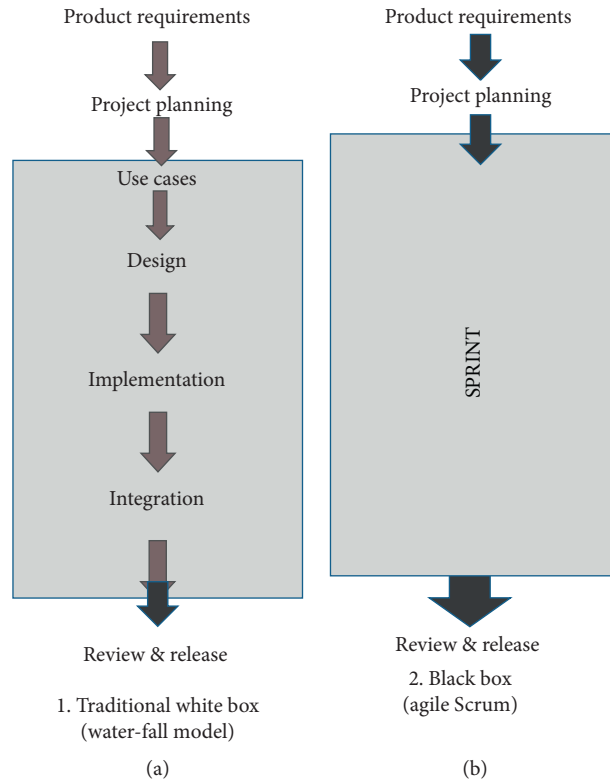


FIGURE 1: Analogy of white box and black box processes.

process. The project development and deployment perspective includes eight core modules that constitute the overall project framework.

This study is organized in seven sections. The current section introduces major terminologies used in this study. The second section presents analysis of existing research in Scrum and SOA approach for an in-depth understanding. The third section is dedicated to studying design and setup, analysis of SOA and Scrum, and integration of Scrum and SOA. The fourth section is about the integration analysis; the fifth section is the discussion about SOA and Scrum integration, while the conclusion and future work of overall study are presented in the sixth and seventh sections, respectively.

2. Literature Study

Agile processes are planned to upkeep early and fast development of software applications. This is made possible by working in iterations [9]. Iterations stress on the delivery of working software that provides value to customers as well as to the project. An iteration is a short development cycle that produces working software and other artifacts which are valuable to both customer and the project [9]. Main concern of agile processes is to provide deliverables and working products in a dynamic way without emphasizing on design models and documentation perspective [7].

Experts of traditional development process conclude that when documentation and analysis process is neglected, it leads to corporate memory loss, especially when complex

and large-scale software are developed. But the experts of agile confirm that, by following agility for software development global software delivery, it has led to better outcomes in gaining concerned objectives [18, 19]. It is clear that organizations will have to face some sort of challenges in coordinating and integrating agility and global service delivery, but also it makes organizations able to deliver quick and error-free software that meets concerned business requirements [20].

Agile software development methodologies are introduced to provide answers to the concerned business community which asks for light-weight, more rapid, and more flexible software development methods [21]. Therefore, agile development methodologies are dedicated for agility, quickness, and eagerness for development [22]. According to Oxford dictionary, agile processes are devoting the importance of being agile, willingness for motion, nimbleness, activity, and dexterity in motion [12]. Some of the agile methodologies like extreme programming and Scrum have acknowledged that customer satisfaction could be achieved through the lightness of concerned processes [12].

Scrum is one of the most used agile management methods. According to last three agile adoption surveys, Scrum highlights empirical feedback, team self-management, and struggling to build thoroughly tested product increments within short iterations [23, 24]. Scrum is an agile methodology which is the most standard way of introducing agility due to its flexibility and straightforwardness and a popular management agile method in industry. Agile software development process facilitates better ways of

developing software by promoting individual work as well as teamwork [11]. Agile processes are planned to maintain early and fast development of software application. This is made possible by dividing the development process into sprints, where sprint stresses on the delivery of working product that provides value to both the project and customer [25]. Agile development of applications in an enterprise surrounding can be challenging because of the compound nature team members and their environments [11].

SOA agrees to the integration of applications, users, and existing system into a flexible architecture that can easily accommodate changes when they are needed in a system [14]. SOA is regarded as one of the best approaches for distributed application development [26]. SOA allows reusing the functionality of existing systems rather than building again from scratch [27]. This feature of reusability in the SOA based applications maximizes economic benefits for the organization. Each service in SOA performs autonomously but is not isolated from the whole system. Each service encapsulates a specific logic in the problem domain. The main features of SOA are reusability, loose coupling, service contract, autonomy, abstraction, discoverability, and statelessness [28].

Researchers and professionals have a mixed opinion about similarity and compatibility of the two approaches. Critics emphasize differences among SOA and agile approaches, arguing that SOA and agile are standing at different development direction: SOA is architecture and agile is a methodology [24, 26]; SOA works in a top-down manner, while agile is inherently a bottom-up approach [29, 30]. SOA is an architectural framework and follows a set of principles, whereas agile is a process model and works at practice level [31]. Based on this view, advocates of SOA and agile integration suggest that one way of combining them is to use SOA framework and guiding principles of agile practices for service development [32, 33]. Some researchers also claim that SOA based systems are developed and deployed differently from traditional Systems [21]. Also, there are many challenges like stakeholders' involvement, business and IT alignment, and reuse of assets. To overcome such type of problems, agility and service orientation are better integrated [34, 35]. It is notable that Scrum and SOA share similar concerns, such as responsiveness to change, new ways of working, flexibility, and business understanding [36].

SOA and Scrum assessment and specification need attention of technology factors along with their relationships to IT related business organization [37] and to individual information consumers. Other requirements such as flexibility and ease of use are defined in terms of such type of system features that can be measured and designed [38, 39].

3. Design of SOA and Scrum Based Environment for IIoT

Scrum as a development process and SOA as an architecture framework deal with similar concerns, for example, flexibility and quick response to change with profound business orientation. Even though this inherent operational similarity

is quite obvious, still not enough research has been made in analyzing the most optimal solution for the integration of these two approaches in order to receive the combined benefits [40, 41]. Also, there is a need to investigate the impact of employing SOA with regard to improving productivity and efficient utilization of time and resources using Scrum methodology.

To investigate this context, a case study has been selected, where Scrum process model is used as a development methodology for the development of an SOA based software system. The SOA based project is named M4S (Mineral resource, Mapping, Modeling, and Management System). The project development and deployment perspective includes eight core modules that constitute the overall project framework. The system is developed following the systematic steps which are discussed in detail in the following paragraphs.

This study is evaluated through identifying, determining, and using one of the best combinations in software engineering, which is designed based on SOA, using highly productive open-source frameworks and tools, development through agile development methodologies, and deployment as a combination of client and cloud applications.

Scrum has been used as an agile process for continuous development and improvement to following changing requirement to refine working prototypes in iterations [42, 43]. The application is designed using SOA to ensure that the case study system modules can be used as independent subsystems through service interface. SOA facilitates exploiting the characteristics and capabilities of both Scrum and SOA for integration, customization, and binding of the desired designed frameworks and tools, the Mineral resource, Mapping, Modeling, and Management System (M4S); ultimately the deployment phase of this system has infrastructure having properties of interactive communication, scalability, and dependability of services.

These deployment requirements are achieved through cloud infrastructure and distributed computing. Modules of the developed application are designed to act as interoperable services and can be used independently and recombined in other modules resulting in an integration of multiple coherent modules. Data banks for mineral, resources, and application to manage those data banks are specifically designed for M4S system. These databanks, libraries, and applications are designed to be used as open-source utilities.

Scrum methodology is used to ensure having a transparent process of analysis and design of M4S and engagement of stakeholders and also ensures a dynamic interaction among the different components of the project. The use of agility and Open-Source System (OSS) principles for the development and project management has permitted targeting a diverse community of stakeholders through multilateral liaison of academia, government, and industry. A collaborative work of these diverse actors played a prime role in M4S development, along with a minor role of individual contributors.

As discussed in ICT R&D project named M4S [<https://www.openm4s.org>], the core framework of M4S utilizing open-source libraries, services, servers, applications, database management system, and an open-source Enterprise resource

planning (ERP) is shown in Figure 2. Client-server architecture and browser-based clients are based on the principles of interoperability and open standards. The figure also shows components and information flow within proposed M4S.

Scrum-driven agile software development methodology is used for managing and organizing the development of modules [15]. Scrum used in the context of this case study along with specified roles and team organization is given in the following section: The process setup is decided according to the specification of agile Scrum. The following is a brief description of roles as these are present and work in a Scrum process environment.

- (a) Scrum master is the one who leads Scrum meetings, maintains the processes of team, and makes sure that the process respects all principles of Scrum [44, 45].
- (b) Scrum coach also facilitates meetings, helps the team to reach consensus, and takes care of quality aspects of the development environment and team.
- (c) Scrum team is a cross-functional team consisting of developers, testers, architects, and analysts. This team is responsible for project results as a whole and is fully involved in development. The team takes care of developing features according to quality and functional requirements [46].
- (d) Product owner is a person who represents the interests of end users and the stockholders. The product owner takes care of quality aspects related to user requirements. A typical sprint cycle is presented in Figure 3.

3.1. Team Organization of M4S Process. Team organization of M4S process is shown in Table 1 for the project development. Product owners provide the roadmap, business case, release plans, and architectural direction. Scrum masters have the knowledge on the architectural framework, usage context of the requirements, and features of M4S. They performed the business analysis and quality assurance tasks, apart from managing the sprint cycles.

Key roles and responsibilities of each member in the project are as follows:

PD (project director) was responsible for planning, monitoring, and managing project activities.

Co-PD (coproject director) was responsible for monitoring project activities, establishing liaison, and coordination with government and industrial partners.

Software and computer systems experts were responsible for directing software architecture design and quality assurance.

Engineering management expert was responsible for directing project management activities and quality assurance.

Team leads were responsible for scheduling and execution of design, development, and deployment tasks.

Developer teams were responsible for development, testing, verification, and documentation of modules.

Coproject director was responsible for maintaining liaison and coordination with government and industrial partners.

Mining and software experts (part time) were responsible for testing and validation of M4S and its modules.

Mentors (student supervisors) were responsible for supervision and guidance to UG and PG students, working on M4S related projects.

Part-time developers were responsible for development, testing, and verification of modules.

Internees were working on various features of M4S as their research projects and developed and documented the assigned projects.

Work breakdown structure (WBS), along with the responsible personnel for each block of activity, of M4S project is shown in Figure 4.

The following are the details of the Scrum process and its different features which are applied in the project.

(a) User stories

A user story captures what the user wants to achieve through the system. This objective is transformed into a user story from the customer's point of view [47]. In accordance with the user stories, acceptance test card is prepared by the product owners (project director and codirector) and the developers implement those stories. The acceptance test card comprises a set of test cases against which the implemented user stories are tested. A user story will be considered complete only if the acceptance tests are satisfactory. Once ready, the user stories are turned into system features [17]. A feature means "work to be done" or a piece of functionality to be implemented in the system.

In M4S project, individual features are decomposed into smaller tasks. Then story cards are designed for each individual task. The completion time for a task was recommended to be from five to fifteen working days, and a working day was normally consisting of eight hours. Normally one or two team members are working on only one story card at a time. Each story or task is assigned priority according to its importance. Team members select stories on their priority base to finish.

Figure 5 depicts two stories that are finished before the selection of next story.

(b) Sprint

According to Scrum process, development is organized into two- or four-week-long periods called sprints. A sprint is a time-boxed development iteration within which selected tasks are implemented [37]. Tasks are assigned to sprints during a sprint planning session at the beginning of each sprint based on the status of the backlog. A one-week sprint (also called sprint 0) is recommended before starting with the M4S project development. This allows the developers to get acquainted with the development

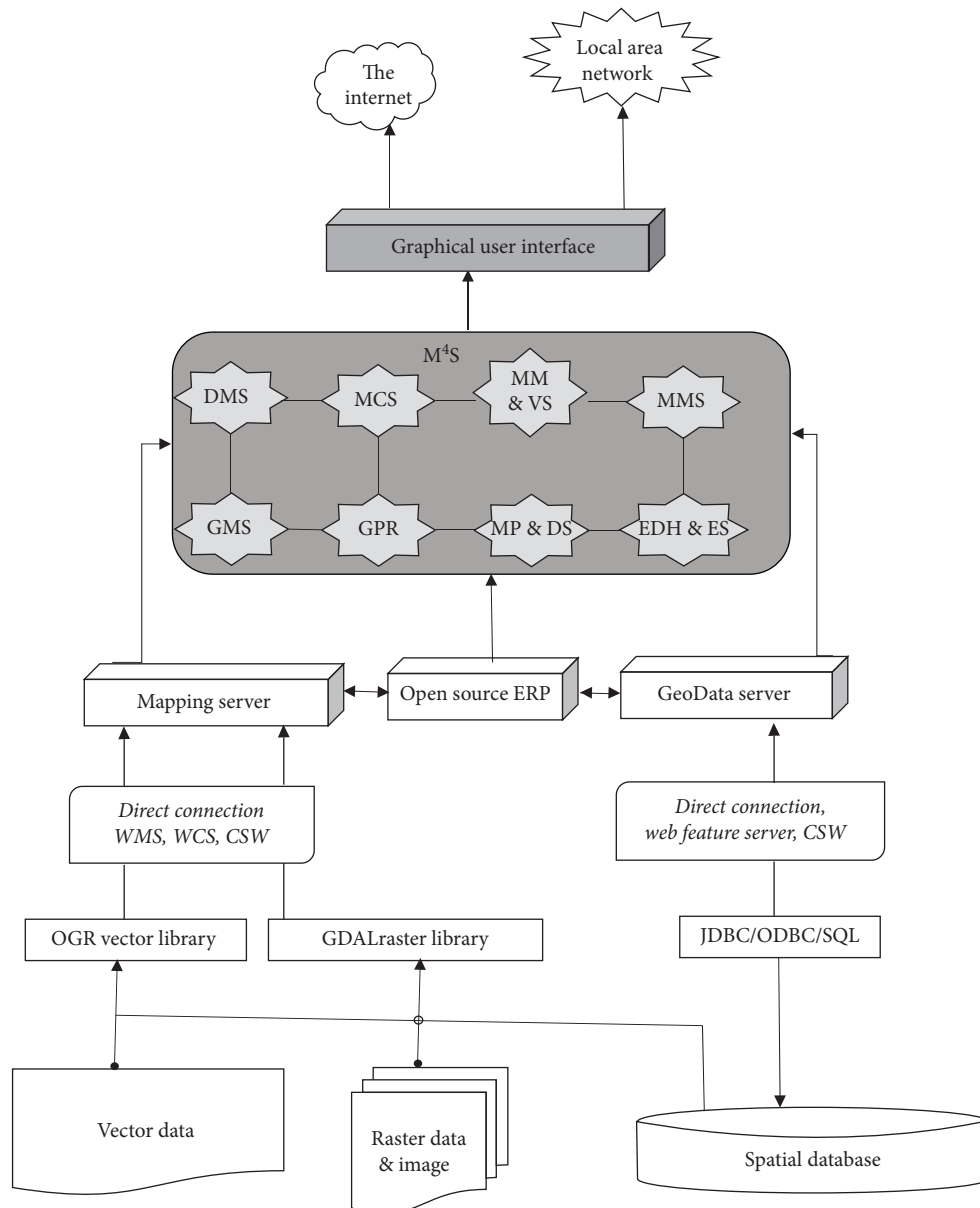


FIGURE 2: Proposed frameworks of M4S [<https://www.openm4s.org>].

environment, technological solutions, and the existing components that are reused.

The first module of M4S is completed in seven sprints. The first sprint consists of total nine stories, in which seven stories were completed within sprint time box, while the remaining two stories were included in the second sprint to be completed next. Each sprint is designed in Microsoft Excel sheet, which contains all the information about each story, team members who worked on a particular story, time required for a particular story to be completed, and so forth. Figure 6 depicts the first sprint as a sample as follows.

Scrum methodology is based on regular meetings where product owner, Scrum team, third parties, and

Scrum master discuss different issues faced/acrued during development time. The meetings are held for the purpose of ensuring smooth running of the project and preserving/maintaining the overall quality phases of the developed system. Scrum process model employs five types of meetings which are daily Scrum meeting, sprint retrospective meeting, sprint review meeting, sprint planning meeting, and sprint release meeting. These five meetings are also held and maintained during the M4S implementation in different situations. These meetings are discussed one by one in the following subsection.

3.1.1. Sprint Planning Meeting. A total of seven sprint meetings have been held to plan the work to be finished in a sprint. Normally a meeting is held after every two weeks.

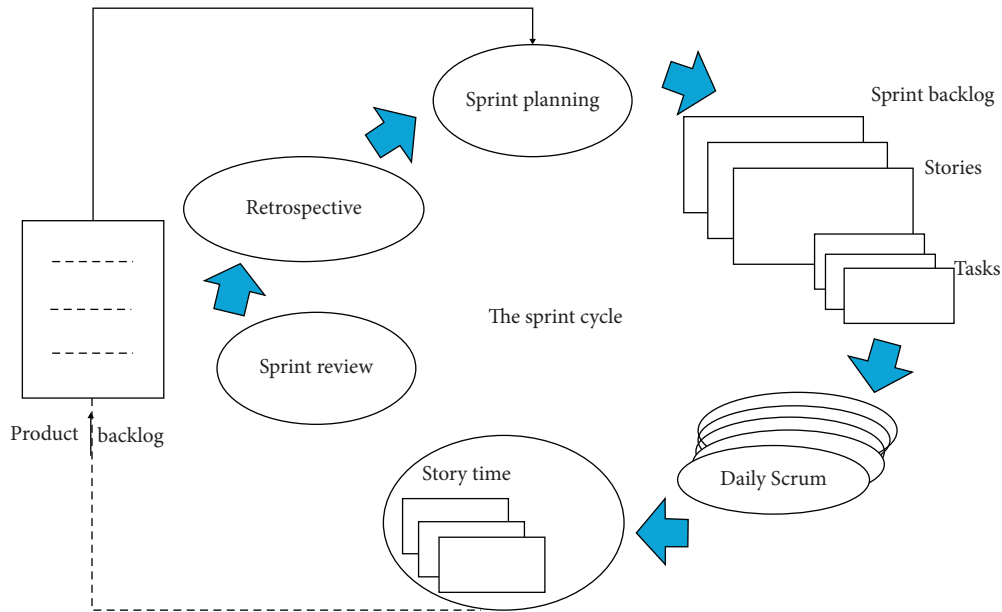


FIGURE 3: Typical sprint cycle.

TABLE 1: Scrum teams for M4S.

Product owner	Product director, coproject director
Scrum master	Team leader
Scrum team	Senior developers Internees (graduate & undergraduate students) Developers and experts (S/W and CS experts, engineering management experts, mining and software engineers, mentors, and part-time developers) integrated in the team for special task like testing and quality assurance Stakeholders and coordinators from industry, government institutions, and academia (analysts)

During this meeting, Scrum team and the product owner (M4S project director & codirector) define the sprint goals. The achievement of the sprint is later evaluated in the sprint review meeting. Product owner works to prioritize the story cards, to describe the highest priority story points to the Scrum team. Finally, Scrum team needs to prepare the sprint plan and sprint backlog that detail the time it takes to do the work [48].

3.1.2. *Daily Scrum.* A daily Scrum meeting is held, the main purpose of which is to make sure that application’s features are being implemented in order according to their planned agenda. Normally this meeting takes place during each sprint in the starting time of work and lasts for approximately 20 to 35 minutes. The meeting is scheduled at exact regular time. During this meeting, each member of the team will answer the succeeding three questions:

- (i) What and how much work have you been doing since yesterday?
- (ii) What are the plans for today work?
- (iii) What are the impediments that would prevent accomplishing the goal?

If any issue occurred, then it is the responsibility of the team leader (formally called Scrum team) to simplify and resolve these complications. Unsolved issues are rediscussed in another meeting. This meeting is held in the same place, at the same time, and for the same duration.

3.1.3. *Sprint Review Meeting.* This meeting is held after each sprint. In this meeting, the team members show what they have done and achieved during the completed sprint. Normally, this meeting takes place in the form of demonstration of the recently developed features. In M4S project, Scrum team has finished at least 95% of product backlog story points fetched to the sprint, and the team achieved most of the sprint goals. In each module of the M4S project, 98% of the features that meet the “done” criteria are accepted and marked as complete. The remaining 2% of the features that are not complete in the specified sprint are rescheduled for future sprints. The project director (product owner) is satisfied by the 98% completion; then the features are closed, and next backlog is updated to include the features that need further development.

- (a) The completed stories for each sprint are accepted by M4S project director and coproject director (product owners).

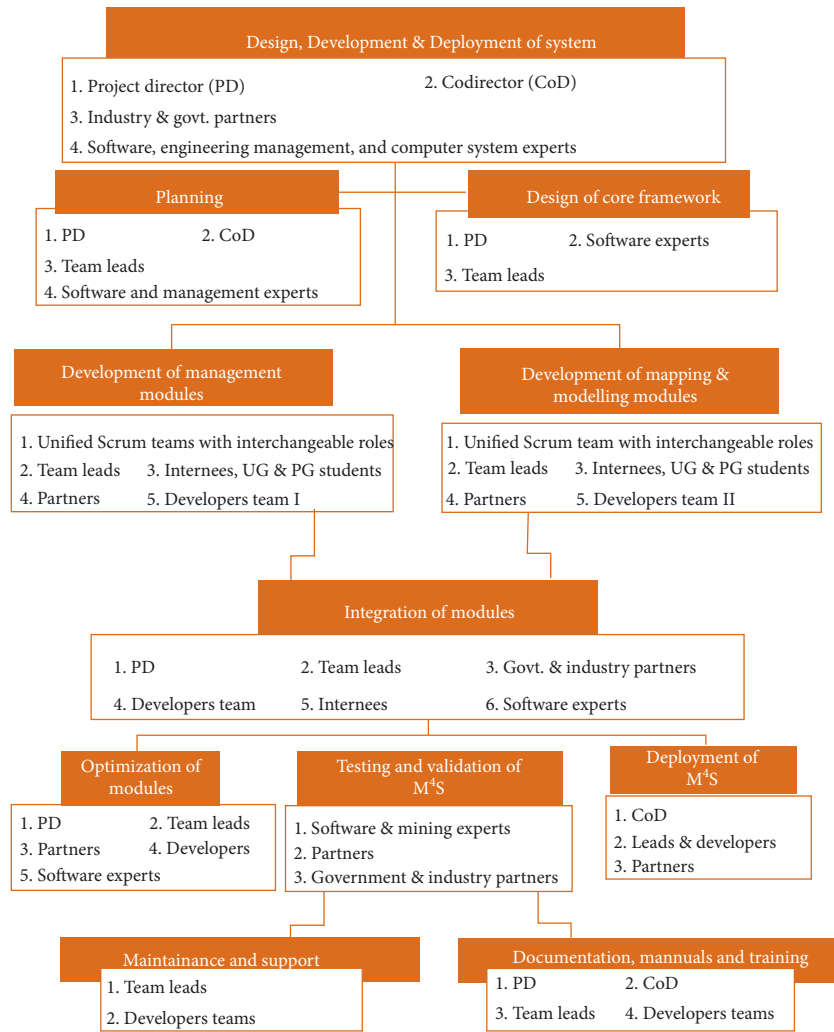


FIGURE 4: Work breakdown structure and resource personnel for M4S project [https://www.openm4s.org].

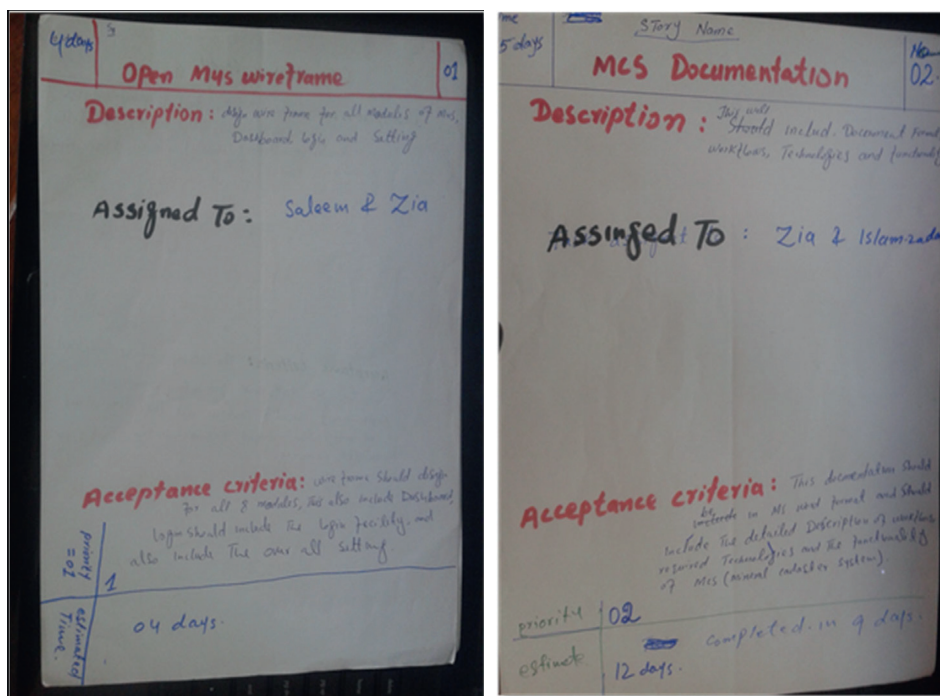


FIGURE 5: Story cards used in the project.

	A	B	C	D	E	F	G
1	M4S Sprint Planning						
2	Sprint #	Sprint:1					
3	Sprint Name	Wireframe & Documentation					
4	Sprint Goal	To build a base for Open M4S, Complete Documentation					
5	Team Member	8	Anwar,Qasim,Zia,Nahyan,Saleem,Zahid,Islam,Iftihar				
6	Sprint Start Date	3/3/2014					
7	Sprint Duration	12	Days				
8	End Date	17/03/2014					
9	Story Name	Tasks Description		Assigned To	Start Date	End Date	Status
10	MCS Documentation	Document Format, Workflows,Technologies,Functionality		Saleem	3/3/2014	17/03/2014	Done
11	Open M4S Wireframe	Wire frame for all modules,Dashboard,login and Settings		Zia	3/3/2014	17/03/2014	Done
12	GeoNetwork Study	Study ,Installation,Knowledge Sharing		Nahyan	3/3/2014	17/03/2014	Done
13	Open Geo	Study ,Installation,Knowledge Sharing		Zia	3/3/2014	17/03/2014	Done
14	Mapping,GIS	Study,Example, Knowledge Sharing		Iftihar	3/3/2014	17/03/2014	Done
15	Sola	Study ,Installation,Knowledge Sharing		Saleem	3/3/2014	17/03/2014	Done
16	SOA	Study ,Installation,Knowledge Sharing		Islam	3/3/2014	17/03/2014	In Progress
17	Wireframe(4 Modules)	Wire frame and documention		Qasim	3/3/2014	17/03/2014	Done
18	ERP Documentation	Study ,Installation,Knowledge Sharing		Zahid	3/3/2014	17/03/2014	In Progress
19							

FIGURE 6: First sprint of M4S project designed in MS excel.

- (b) The finished features are accepted by project director during the meeting.
- (c) All the acceptance tests run for the stories in concerned sprint.
- (d) The whole code has been passed through code reviewing process.
- (e) No critical bugs are found in the bug backlog.
- (f) All finished story points for release are accepted.
- (g) The products are tested properly and have not found any serious bugs.
- (h) A successful backup is made.
- (i) Most of the deployment documents are up to date.

3.1.4. *Sprint Retrospective.* In this meeting, M4S team members reread the approach team workings, interrelations, behavioral features, and refining methodological skills, so that the ensuing sprint is more rapid, and so forth. Scrum team and team leader (Scrum master) discussed three things: what went well, what did not go well, and what improvements can be considered in the next sprint. The overall performance of the team is evaluated (in the form of metrics) and improvement strategies (in the form of KPIs) are identified, which are adjusted in our previous article. It is suggested that team members should have to share technical skills and knowledge, should keep in continuous communication with product owner, and so on. This meeting is held at the end of each sprint and lasts approximately two to three hours.

3.1.5. *Release Planning.* A release planning meeting is a continuous and uninterrupted process of describing,

prioritizing, and splitting the system features in a release backlog. This involves the identification and commitment on the following:

- (a) Release goals.
- (b) Prioritized set of user stories (features to be developed).
- (c) Rough estimates of user stories.
- (d) Release dates.

The main input steps that are held during this meeting in M4S project are the following:

- (a) Team members estimate the user stories, i.e., make rough estimates of the relative size of the stories.
- (b) Establish velocity, i.e., determine how many story points are completed in each sprint; the details are given in our other article.
- (c) Compute forecast, i.e., date-based release is estimated to complete (velocity × number of sprints) story points. Functionality-based releases are estimated to complete in (total story points ÷ velocity) sprints.

The highest priority stories, whose sum is no more than the number of story points (computed as mentioned above), are selected. For a functionality-based release, if the estimated completion date (computed above) is acceptable, all the stories are selected for the release.

4. SOA and Scrum Integration

This section analyzes the integration of Scrum, a software development process, and SOA, an architectural style. SOA

introduces an organized and well-defined software design, where change is supported through principles of agility along with the application of design patterns and standards to achieve the quality, efficiency, and productivity [9]. Service reusability and abstraction are applied for designing flexible and adaptable systems [16]. SOA uses agility at design level as new services are designed and old ones are evolved on the underlying architecture and not by changing the architecture itself. Scrum allows development in increments, facilitating faster feedback among customer(s) and development team. Scrum and SOA both support the corporate process that promotes setting up aligned businesses and development strategies.

The analysis of Scrum and SOA process in M4S project development environment showed that Scrum and SOA use similar principles to reach the joint goal of required software development. Even though SOA runs in an organized and well-controlled setup, it can benefit from agile Scrum at the same time, without any unsuited overlapping. A clear strategy to introduce change in increments should, however, be designed.

Use of Scrum for traditional applications or conventional architectures is also possible by describing requirements into a backlog and incremental deliverables for the customers. Nevertheless, it is expected to get increased developmental efforts and overall project cost [49].

To get proposed benefits and maximum output through an agile process, it should be kept in mind that the underlying software architecture and IT organization environment also support the process in terms of responsiveness and costs. Otherwise, the organization may end up getting only minimal benefit from the agile process.

As already discussed, SOA and Scrum are approaches that follow different directions. In the services development scenario, SOA approach follows from top-down approach (services are building in the top of SOA system), while Scrum follows bottom-up approach (starting from initial planning to prototype delivery) as using a process development methodology. The following question arises: how are these different approaches compatible with each other when employed together for a development process? Another question is, does SOA also follow the agility just like Scrum process? If the answer is yes, then how? Finally, how could these two approaches be integrated with each other to get benefits offered by both individually? This section refers to these questions and also discusses SOA and Scrum metrics having commonalities. The SOA and Scrum typical developmental framework is depicted in Figure 7.

Table 2 identifies most important metrics of SOA and Scrum, which can provide more value to business and to those whose aim is to use SOA and Scrum together in a software development venture.

Some of Scrum and SOA metrics are used for common purpose, which are integrated to be used for Scrum and SOA combination. These metrics are integrated from Scrum and SOA metrics which are used for the same purpose but using different terminologies. Through these metrics, development process can be streamlined when a

proper measure is taken, which will provide more business agility, flexibility, and compatibility for SOA and Scrum environment.

The “completed stories vs. planned stories” metric of Scrum and “new service created and used as percentage of total service” metric of SOA measure the production of product using ratio and percentage as a scale device. They both are used for service or work measurements but using different terminologies. Also, the three metrics which are team velocity, development time, and average development time to develop a service measure the team progress in terms of sprint and time required for a service which is to be completed in a particular sprint within time. Due to common goals of these metrics, these could be combined into team velocity metrics which will be considered as a metric for SOA and Scrum integration. For quality measurement, Scrum and SOA used separate metrics, but the purpose was the same; therefore these two metrics can be used as quality assurance metric for measuring the services’ quality using Scrum process model.

The team enthusiasm metric measures whether the team members are happy and work eagerly or not. When the team members are happy and work in a comfortable environment, they will work willingly and will follow the free-planned architecture policies and rules. They will work in a collaborative environment. The communication between team members will always be positive when they work eagerly due to happiness.

5. Discussion of Compatibility of Scrum and SOA

The main purpose of SOA and Scrum is to make the whole enterprise agile by using services as the building blocks for software applications [50]. Also software development through Scrum process model means to increase organization agility by bringing together Scrum practices that could increase communication, collaboration, and feedback [13, 14]. As already discussed, Scrum and SOA are generally viewed with similar concerns, but still there exist some diversities and incompatibility issues between the two approaches. The compatibility and commonalities of these two approaches are discussed in order to make a ground for the integration of both development approaches.

5.1. Compatibility of Scrum and SOA. It may sound confusing as to why there is a need to find similarities and compatibilities among Scrum and SOA. Scrum and SOA are two different paradigms as one is a process and the other is an architectural style. Still, it is logical and relevant to find compatibilities of both when used together, for example, to develop an SOA based system using Scrum as a software development methodology, while most SOA teams are subconsciously aware of the way of design and development of services. The whole focus revolves around service design policies. SOA’s nature inspires specific team makeup and style of communication within teams according to policies just like Scrum practices [15, 51]. It can be said that Scrum is

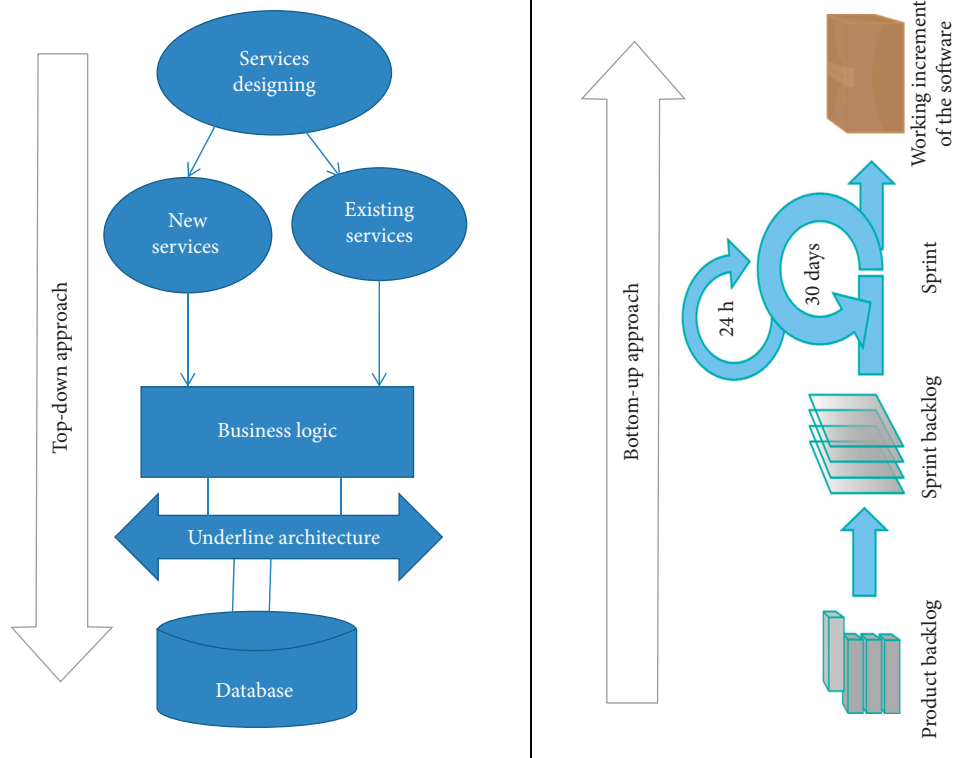


FIGURE 7: Typical developmental framework of SOA and Scrum.

like the human hands that work using gloves, while SOA is like that glove. It is understood that most of Scrum and SOA principles are not in conflict with each other because they both are adaptive approaches [2]. Meanwhile SOA with a traditional approach, for example, waterfall model, becomes predictive, as predictive methods completely depend on the requirement analysis and careful planning at the beginning of the cycle. Any change that is to be included will go through a strict change control management and prioritization.

The agile model uses an adaptive approach where there is no detailed planning and only clear future tasks are those related to the characteristics that must be developed. The team adapts to dynamic changes in the product requirements. The product is frequently tested, minimizing the risk of major faults in the future. Interaction with the clients is the strong point of agile methodology and open communication and minimal documentation are typical characteristics of the agile development environment. Teams collaborate closely and are often located in the same geographical space. Therefore, an agile model is better suited for the rapid development of adaptive services in SOA projects development. Application development through Scrum without a clear and strong idea of the aims of the organization will be useless. SOA without a clear image of how exactly to design and build using Scrum process model rules is a waste of resource and time. Also using SOA, as a pervasive strategy, for developing software application is increasing, since it focuses on the ability to respond to changes. Since adapting to changes is the indispensable

concept of SOA development as well as agile development, it seems that using agile methodology is a natural fit to develop SOA applications.

Hence, Scrum and SOA are about agility that can be used together by applying a number of rules and principles, which are not in conflict with each other. This way they maintain each other in balance.

5.2. Scrum and SOA Commonalities. As already discussed, SOA is an architectural approach that focuses on the fact that business organizations must be intelligent and could respond to rapid changes in business. By developing services, one can reach closer to the indefinable goal of software reuse. In SOA approach, different teams build individual services and then these services are integrated in an application. [49, 52]. Meanwhile Scrum is an agile process model for application development, which stresses on responding to changes [8]. By introducing both technical and nontechnical practices, teams are able to support businesses to become agile. Scrum and SOA approaches are paired by nature and share general objectives. In both cases, change is adaptable, and organizations need to effectively cope with that change. Therefore, Scrum could be a choice when building SOA based applications.

5.3. Adversity of SOA and Scrum. Although Scrum and SOA are compatible, still there are some major distinctions among the two approaches, which should be understood and handled while working with SOA and Scrum together in a

TABLE 2: Scrum and SOA common metrics.

	Scrum	SOA	Commonalities
1	Completed stories versus planned stories	Newly created & used as a percentage of total services	The main purpose of these two metrics is to measure the ratio and percentage of completed work (services) developed in one sprint.
2	Team velocity	Development time & average development time to develop a service	These three metrics measure the team progress in terms of sprint and time required for a service which is to be completed in a particular sprint within time. So these metrics could be combined into team velocity metric which will be considered as a metric for SOA and Scrum integration.
3	Quality delivered to customer	Service quality assurance	The aim of these metrics is to measure the service quality when applying the Scrum development process model. The quality is a common feature for both metrics which can be combined to make a metric for SOA and Scrum integrations measurements.
4	Team enthusiasm	Violation of architecture policies	When the team members are happy and satisfied and work in comfortable environment, then they will communicate with each other collaboratively and in a friendly way. They will have full attention and focus on product development through this product quality will remain standard. Also they will willingly follow the preplanned architecture policies and rules. When the Scrum team is happy and in restful environment, then they can develop a large number of services of high quality in small amount of time. So we can say that the “team enthusiasm & communication” metrics of Scrum and “violations of architecture policies” & “average time to service development” metrics are dependent on each other; these can have an effect on project when these are not concentrated. These metrics are used to measure the behavior of how they follow rules and policies during development environment.
5	Team communication	Average time to service development	
6	Retrospective process improvement	Service accessibility & usability	These two metrics can be integrated together to represent a common metric for both Scrum and SOA. Because the retrospective meeting is held in the last of all practices of Scrum in which the overall activities could be revived, when the services are developed in a sprint, a review meeting will be arranged in which we can test the developed service functionality and usability of how to access the service and how it works.
7	Technical debt management	Reduction in the project and maintenance expense	The main purpose of these two metrics is to reduce the product development cost through best management and utilization of resources and team member’s skills. These two metrics can be integrated in one common metric for combined use of Scrum and SOA approaches.

project [53]. One of the leading causes is that the two approaches have different origins and diverse directions. Scrum is traditionally small-project-based, although with process improvement and experience of practitioners have gain knowledge and learned experience to adapt the rules of the Scrum manifesto can also be applied to large software development projects. SOA is a top-down and divide-and-conquer approach to applications development. The “divide” part naturally results in low communication between teams. There are three main areas where Scrum and SOA clash with each other.

- (1) SOA encourages that architecture be designed upfront, while in Scrum community big design upfront (BDUF) is considered as an antipattern.
- (2) SOA encourages teams to split along functional lines, while Scrum encourages cross-functional teams.
- (3) SOA does not have any formal feedback and communication among development teams, while Scrum

is focused on frequent feedback at both a technical level and personal level.

When using SOA in a large setup, it may be complicated to effect any change [54]. There may also be other issues beyond software agility and cost, for example, lack of team communication and discoverability of services. So instead we should strive for closer cooperation and openness among teams. Each team will make available an easy modifiable version of its service for internal test. The service team will separate test data from service, so it survives even if the code is messed up. The service team will provide clean and simple documentation on how to add new calls to the service stub.

6. Conclusion

Service-oriented architecture (SOA) allows reusing the functionality of existing systems rather than building again from scratch. This feature of reusability in the SOA based

applications maximizes economic benefits for organizations. Scrum process model, on the other hand, tends to focus on iterations and client suggestions to improve performance and allow for the predictability of varying requirements. The study analyzes a basis for identifying commonalities and compatibilities in Scrum and SOA process to achieve maximum benefits of the organized Scrum management process for SOA based applications development. The study establishes that most of the Scrum and SOA principles are not in conflict with each other. Both Scrum and SOA are about agility which can be applied using rules and principles that do not clash with each other. To confirm this compatibility, the performance of an integrated Scrum and SOA development environment can be tested through formal KPIs based on the individual Scrum and SOA metrics commonly used by Scrum and SOA practitioners in the software development industry. These KPIs will provide a formal approach to measure agility, complexity, efficiency, and value of Scrum and SOA for those teams who want to use the Scrum and SOA in an integrated environment. Although the output of the study is complete, the researchers identify some limitations which should be considered before applying the results for an industrial Scrum and SOA integrated project.

7. Future Work

The following are some of the important points for future work based on the outcome of this research:

- (1) To find out what will be the impact of Scrum and SOA integration on the project performance if used in a distributed environment.
- (2) To define a formal process for the evaluation of identified metrics and KPIs in nondistributed and distributed Scrum and SOA integrated projects.

Data Availability

No data are available.

Conflicts of Interest

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

References

- [1] N. Bieberstein, M. Fiamante, K. Jones, S. K. Bose, and R. Shah, *Service-Oriented Architecture Compass: Business Value, Planning, and Enterprise Roadmap*, FT Press, Indianapolis, IN, USA, 2006.
- [2] M. Lankhorst, *Agile Service Development: Combining Adaptive Methods and Flexible Solutions*, Springer Science & Business Media, Berlin, Germany, 2012.
- [3] F. Rago, "Self-organizing business networks, SOA and software maintenance," in *Proceedings of the International Multi Conference of Engineers and Computer Scientists*, Hong Kong, China, March 2008.
- [4] A. Manifesto, "Agile manifesto," *Haettu*, vol. 14, p. 14, 2012.
- [5] R. Kumar, P. Maheshwary, and T. Malche, "Inside agile family software development methodologies," *International Journal of Computer Sciences and Engineering*, vol. 7, no. 6, pp. 650–660, 2019.
- [6] V. Jiménez, P. Afonso, and G. Fernandes, "Using agile project management in the design and implementation of activity-based costing systems," *Sustainability*, vol. 12, no. 24, p. 10352, 2020.
- [7] S. Mohanarajah and M. A. Jabar, "An improved adaptive and dynamic hybrid agile methodology to enhance software project success deliveries," *Journal of Theoretical & Applied Information Technology*, vol. 75, no. 3, pp. 301–325, 2015.
- [8] I. Zada, S. Shahzad, and S. Nazir, "Issues and implications of scrum on global software development," *Bahria University Journal of Information & Communication Technology*, vol. 8, no. 1, 2015.
- [9] N. Naik and P. Jenkins, "Relax, it's a game: utilising gamification in learning agile scrum software development," in *Proceedings of the 2019 IEEE Conference on Games (CoG)*, IEEE, London, UK, August 2019.
- [10] P. Kutschera and S. Schäfer, "Applying agile methods in rapidly changing environments," in *Proceedings of the 1st RTO Symposium on Technology for Evolutionary Software Development*, Citeseer, Bonn, Germany, September 2002.
- [11] M. Awad, *A Comparison between Agile and Traditional Software Development Methodologies*, University of Western Australia, Perth, Australia, 2005.
- [12] W. M. Theunissen, A. Boake, and D. G. Kourie, "In search of the sweet spot: agile open collaborative corporate software development," *ACM International Conference Proceeding Series*, vol. 150, pp. 268–277, 2005.
- [13] S. W. Ambler, *The Agile Scaling Model (ASM): Adapting Agile Methods for Complex Environments*, IBM Corporation, Armonk, NY, USA, 2009.
- [14] S. W. Ambler and M. Lines, *Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise*, IBM Press, Indianapolis, IN, USA, 2012.
- [15] A. Bhagwat, "Role of beacon architecture in mitigating enterprise architecture challenges of the public sector," in *Advances in Government Enterprise Architecture*, p. 502, IGI Global, Hershey, PA, USA, 2008.
- [16] H. Kerzner, *Project Management: A Systems Approach to Planning, Scheduling, and Controlling*, John Wiley & Sons, Hoboken, NJ, USA, 2017.
- [17] D. Gabioud, E. László, G. Basso et al., *D3.2–Overall System Requirements and Functional Specifications*, Scalable Energy Management Infrastructure for Aggregation of Households (SEMIAH) project, 2015.
- [18] C. Verma, *The Effects of Organizational Culture on Risk Management during Software Development*, The British University in Dubai (BUiD), Dubai, UAE, 2009.
- [19] G. Borrego, A. L. Morán, R. R. Palacio, A. Vizcaíno, and F. O. García, "Towards a reduction in architectural knowledge vaporization during agile global software development," *Information and Software Technology*, vol. 112, pp. 68–82, 2019.
- [20] J. McGovern, O. Sims, A. Jain, and M. Little, *Enterprise Service Oriented Architectures: Concepts, Challenges, Recommendations*, Springer Science & Business Media, Berlin, Germany, 2006.
- [21] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, "Agile software development methods: review and analysis," 2017, <http://arxiv.org/abs/1709.08439>.

- [22] M. Singh, N. Chauhan, and R. Popli, "A technique for transitioning of plan driven software development method to distributed agile software development," in *Proceedings of the International Conference on Innovative Computing & Communications (ICICC) 2020*, Delhi, India, February 2020.
- [23] M. Perkusich, L. Chaves e Silva, A. Costa et al., "Intelligent software engineering in the context of agile software development: a systematic literature review," *Information and Software Technology*, vol. 119, p. 106241, 2020.
- [24] S. Moyo and E. Mnkandla, "A novel lightweight solo software development methodology with optimum security practices," *IEEE Access*, vol. 8, pp. 33735–33747, 2020.
- [25] M. S. Nazare, *Evaluation of Agile Software Development Methodologies and its Applications*, California State Polytechnic University, Pomona, CA, USA, 2019.
- [26] O. Philipp, K. Stefan, and S. Eric, "Model-based resource analysis and synthesis of service-oriented automotive software architectures," in *Proceedings of the 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS)*, IEEE, Munich, Germany, September 2019.
- [27] O. Goubali, A. Idir, L. Poinel, L. Boulhic, D. Kesraoui, and A. Bignon, "Service-oriented control-command components for designing complex systems," in *Proceedings of the International Conference on Human-Computer Interaction. 2019*, Springer, Orlando, FL, USA, July 2019.
- [28] M. Rehan and G. A. Akyuz, "Enterprise application integration (EAI), service oriented architectures (SOA) and their relevance to e-supply chain formation," *African Journal of Business Management*, vol. 4, no. 13, pp. 2604–2614, 2010.
- [29] Y. Baghdadi, "A comparison framework for service-oriented software engineering approaches," *International Journal of Web Information Systems*, vol. 9, no. 4, pp. 279–316, 2013.
- [30] F. Kamoun, "The convergence of business process management and service oriented architecture," *Ubiquity*, vol. 8, no. 14, p. 1, 2007.
- [31] Z. Dragičević and S. Bošnjak, "Agile architecture in the digital era: trends and practices," *Strategic Management*, vol. 24, no. 2, pp. 12–33, 2019.
- [32] J. T. F. Chaves, "Service-oriented architecture (SOA), agile development methods and quality assurance (QA): a case study," Master's Thesis, University of Brasília, Brasília, Brazil, 2019.
- [33] S. Abidi, M. Fakhri, M. Essafi, and H. Ben Ghazela, "A comprehensive framework for evaluating web services composition methods," *International Journal of Web Information Systems*, vol. 15, no. 3, pp. 324–345, 2019.
- [34] M. Fischer, F. Imgrund, C. Janiesch, and A. Winkelmann, "Directions for future research on the integration of SOA, BPM, and BRM," *Business Process Management Journal*, vol. 25, no. 7, pp. 1491–1519, 2019.
- [35] A. Plugge, S. Nikou, and H. Bouwman, "The revitalization of service orientation: a business services model," *Business Process Management Journal*, 2020.
- [36] J. Paramanathan, "Security of lightweight-and heavyweight-IT in a high-tech hospital," Master Thesis, University of Oslo, Oslo, Norway, 2019.
- [37] M. Paasivaara, S. Durasiewicz, and C. Lassenius, "Using Scrum in distributed agile development: a multiple case study," in *Proceedings of the 2009 Fourth IEEE International Conference on Global Software Engineering*, IEEE, Limerick, Ireland, July 2009.
- [38] M. Farsi, M. Badawy, N. Abdelmoneim, H. Arafat Ali, and Y. Abdulazeem, "QoS-aware framework for performance enhancement of SOA in enterprise IT environments," *IEEE Access*, vol. 7, pp. 107699–107715, 2019.
- [39] J. Bloomberg, *The Agile Architecture Revolution: How Cloud Computing, Rest-Based SOA, and Mobile Computing Are Changing Enterprise IT*, John Wiley & Sons, Hoboken, NJ, USA, 2013.
- [40] C. Ardito, M. T. Baldassarre, D. Caivano, and R. Lanzillotti, "Integrating a SCRUM-based process with human centred design: an experience from an action research study," in *Proceedings of the 2017 IEEE/ACM 5th International Workshop on Conducting Empirical Studies in Industry (CESI)*, IEEE, Buenos Aires, Argentina, May 2017.
- [41] H. Yang, *A Quest to Solve Information System Agility Problems: A SaaS Experience*, Victoria University of Wellington, Wellington, New Zealand, 2018.
- [42] A. Alhazmi and S. Huang, "Integrating design thinking into scrum framework in the context of requirements engineering management," in *Proceedings of the 2020 3rd International Conference on Computer Science and Software Engineering*, Beijing, China, May 2020.
- [43] S. Al-Saqqa, S. Sawalha, and H. AbdelNabi, "Agile software development: methodologies and trends," *International Journal of Interactive Mobile Technologies*, vol. 14, no. 11, p. 246, 2020.
- [44] K. S. Rubin, *Essential Scrum: A Practical Guide to the Most Popular Agile Process*, Addison-Wesley, Boston, MA, USA, 2012.
- [45] K. Schwaber, *Agile Project Management with Scrum*, Microsoft Press, Redmond, WA, USA, 2004.
- [46] G. R. King, "Using the agile development methodology and applying best practice project management processes," Master's Thesis, Naval Postgraduate School, Monterey, CA, USA, 2014.
- [47] C. O'hEocha and K. Conboy, "The role of the user story agile practice in innovation," in *Proceedings of the International Conference on Lean Enterprise Software and Systems*, Springer, Helsinki, Finland, October 2010.
- [48] K. Beck, J. V. Sutherland, K. Schwaber et al., *The Agile Manifesto*, 2001.
- [49] A. Andriyanto and R. Doss, "Problems and solutions of service architecture in small and medium enterprise communities," 2020, <http://arxiv.org/abs/2004.10660>.
- [50] Z. Stojanović and A. Dahanayake, *Service-Oriented Software System Engineering: Challenges and Practices*, IGI Global, Hershey, PA, USA, 2005.
- [51] X. Wang, *Agile and Lean Service-Oriented Development: Foundations, Theory, and Practice: Foundations, Theory, and Practice*, IGI Global, Hershey, PA, USA, 2012.
- [52] S. Graham, G. Daniels, D. Davis et al., *Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI*, SAMS Publishing, Indianapolis, IN, USA, 2004.
- [53] A. S. Dadras, "IT agility through service-oriented architecture" Ph.D. Thesis, The University of New South Wales, Kensington, Australia, 2016.
- [54] A. Kumar, R. Liu, and Z. Shan, "Is blockchain a silver bullet for supply chain management? technical challenges and research opportunities," *Decision Sciences*, vol. 51, no. 1, pp. 8–37, 2020.

Review Article

A Review on Multicriteria Decision Support System and Industrial Internet of Things for Source Code Transformation

Qinxia Hao ^{1,2}, Shah Nazir ³, Xiaoxu Gao,⁴ Li Ma,¹ and Muhammad Ilyas⁵

¹School of Communication and Information Engineering, Xi'an University of Science and Technology, Xi'an 710054, China

²School of Safety Science and Engineering, Xi'an University of Science and Technology, Xi'an 710054, China

³Department of Computer Science, University of Swabi, Swabi, Pakistan

⁴School of Energy Engineering, Xi'an University of Science and Technology, Xi'an 710054, China

⁵Department of Computer Science and IT, University of Malakand, Malakand, Pakistan

Correspondence should be addressed to Qinxia Hao; hao_qinxia@sina.com and Shah Nazir; snshahnzr@gmail.com

Received 19 December 2020; Revised 30 December 2020; Accepted 7 January 2021; Published 13 January 2021

Academic Editor: Sikandar Ali

Copyright © 2021 Qinxia Hao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The large scale increase of communication and number of devices in the Industrial Internet of Things (IIoT) has rapidly enabled practitioners to make decisions based on multicriteria. Multicriteria decision support systems (MCDSSs) play an important role in decision-making for a particular situation based on several criteria. Making of decision based on multicriteria is the main issues for research community and practitioners of the IIoT. Several decision support systems (DSSs) are offered for making decisions which have the potentiality to support the activities of the decision-making process. The suggested study shows a review on the existing decision support systems for the IIoT for source code transformation which will enable research community and practitioners of the industry to use the existing methods, tools, approaches, and techniques and to provide novel solutions for the smooth industry of Internet of Things.

1. Introduction

The expansion in the communications and increase of smart devices in the Industrial Internet of Things (IIoT) has speedily empowered practitioners and researchers for making decisions according to several criteria. With the passage of time advancements in communication of diverse smart devices in network, rise in population Sensors, actuators, IoT, and others, size of data is increasing. Recent methodologies and approaches on the way to solving issues of the increase of data into diverse natures including value, volume, variety, velocity, and veracity, extracting significant information are challenging issues, DSS is furnished with the control of MCDSS for supporting the decisions maker in right decision in complex situations. Based on the success of MCDSS, practitioners and researcher are endeavouring to incorporate the control of intelligent decision-making for the available alternative [1–3].

DSSs are used in diversity of domains and applications for supporting the decision maker in taking right and

appropriate decision. The DSS applications are evidenced in different areas of life. Such applications include the DSS in agriculture, business, energy, and so on [4–15]. Numerous domains and applications have explored methods and theories regarding decision-making for alternatives including from simple to advance, intelligent, and smart systems [16, 17]. DSS is a dynamic field of research where studies find new approaches for evaluating various criteria, proposing frameworks which are intelligent and robust for improving the potentiality of DSS. Owing to the success of DSS in the process of decision-making, researchers are trying to provide more trustworthy, effective, and robust mechanisms for solving the current as well as the upcoming issues. MCDSS plays an important role in the process of decision-making for a particular situation based on diverse criteria. Decision-making based on multicriteria is the main issue for practitioners and researchers in the area of IIoT. Source code transformation is done for various purposes such as to optimize the efficiency of the source code,

minimize the source code, and mostly to hide the identity of source code. Various preservation techniques are used for changing or modifying the source code [18–22]. Practitioners are mostly embedding the watermark or digital signature for protecting the identity and ownership of the source code. But due to the advancement of information technology, such watermark and digital signature can be removed. Several DSSs are available for making decisions which have the potentiality for supporting the activities of decisions-making and provide solutions for it. A comprehensive review of the existing approaches should be presented for showing the approaches, tools, and techniques practiced.

The proposed study presents an overview on the available decision support systems for IIoT which will enable practitioners and researchers to study the present methods, tools, approaches, and techniques and to provide novel solutions for the smooth industry of IoT. The study provides a comprehensive report of the existing approaches of DSS used for IIoT.

The paper is organised as follows: Section 2 shows the related work of the existing literature to the proposed study and the approaches for Industrial Internet of Things.

2. Approaches for IIoT

Diverse solutions of approaches, techniques, methods, and tools of decision support systems are devised by researchers to tackle different situations of IIoT. These solutions consist of dealing the problems from simple to more complex with various criteria of handling. Liu et al. [23] proposed a framework of IIoT cloud-fog hybrid network for data processing of industry. Based on the experimental results, it was revealed that the planned framework can reduce delay processing of industry data effectively. Sahal et al. [24] studied the strong point and flaws of open source technologies for big data and stream processing to setup its applications for use cases of industry 4.0. Khan et al. [25] offered the idea of IIoT in a novel manner for supporting readers to comprehend the IIoT. Studies presented for research in the area of IIoT are reported and shown. The research highly spots the empowering technologies for the IIoT and the issues to the IIoT. Gulati and Kaur [26] analysed the main opportunities assimilated from the idea of IoT into industry with suggesting reference architecture. A model of ontology was planned to propose the model from a semantic perception. For the relationship management, a method among industrial resources was offered.

Aceto et al. [27] elaborated the comprehensive explanation of the main approaches and technologies used in support of Healthcare 4.0, benefits, the key scenario applications, multidisciplinary issues, and the derivations. Rehman et al. [28] examined the existing technologies of big data analytics, algorithms, and strategies that can rapid the progress of perceptive IIoT framework. Important factors from the literature including applications of industrial analytics, types of analytics, analytics techniques, analytics tools, requirements, and

sources of data were classified and characterised. The frameworks and case studies of the diverse accomplishments were reported that have turn a profit by BDA. Ordieres-Meré et al. [29] examined the functional properties and structure of cloud manufacturing and planned a business intelligent architecture for empowering dispensing pertinent KPIs recognized with intrigued process data with the support layer of dependability. The authors [30] offered the concept of IoT data management, the literature associated with data IoT management, and the related way-out and identified challenges of open research.

Alexopoulos et al. [31] proposed the architecture of IIoT, and its detail of expansion to help the service of industrial product system life cycle. Mobile phone services are used as mobile computing in the IoT with mobile apps or through M-Health care system [32]. Younan et al. [33] offered a research with a wide-ranging analysis of the current issues in the literature and recommended the use of technologies for allowing the data study and search in upcoming IoT search engines. They offered two case studies for showing encouraging development on intelligence and smartness of IoT presentations based on the incorporation of information and communication technologies. The smart phone applications were presented for the identification of patients' diseases in the fields of gynaecology and paediatrics [34]. Ge et al. [35] showed a report on the big data technologies in diverse IoT domains for enabling and motivating distribution of knowledge through the domains of IoT. The similarity and difference among the big data technology in dissimilar domain with the technology reusability in the IoT domain were presented. Souza et al. [36] proposed digital twin architecture design guidelines through the incorporation of existing technologies and IIoT.

3. Analysis of the DSS for IIoT Based on Popular Libraries

DSS plays a significant role in every field of life. With the advancements of technology and information communication, decision-making on right and appropriate time is a challenging issue for the industrial Internet of things. Taking decision on the right and appropriate time can ultimately lead the industry into success. Diverse approaches have been practiced in order to provide solutions for different situations of the IIoT. Jiang [37] offered a method which initially studies the developments of IoT, technologies associated with smart cities and cloud computing and then emphasized on technology of IoT and cloud computing. Urquhart and Mcauley [38] offered a method for risks lying for IIoT drawn both on the perspectives of technical and regulatory. Humayun et al. [39] showed a wide-ranging description of the growth, prevention, and moderation of Ransomware in the background of IoT. Dachyar et al. [40] presented a comprehensive detail of the 26420 articles published in the IoT field. Gierej [41] offered the concept of the business model for corporations applying technologies of the IIoT. The model is established to support traditional companies in

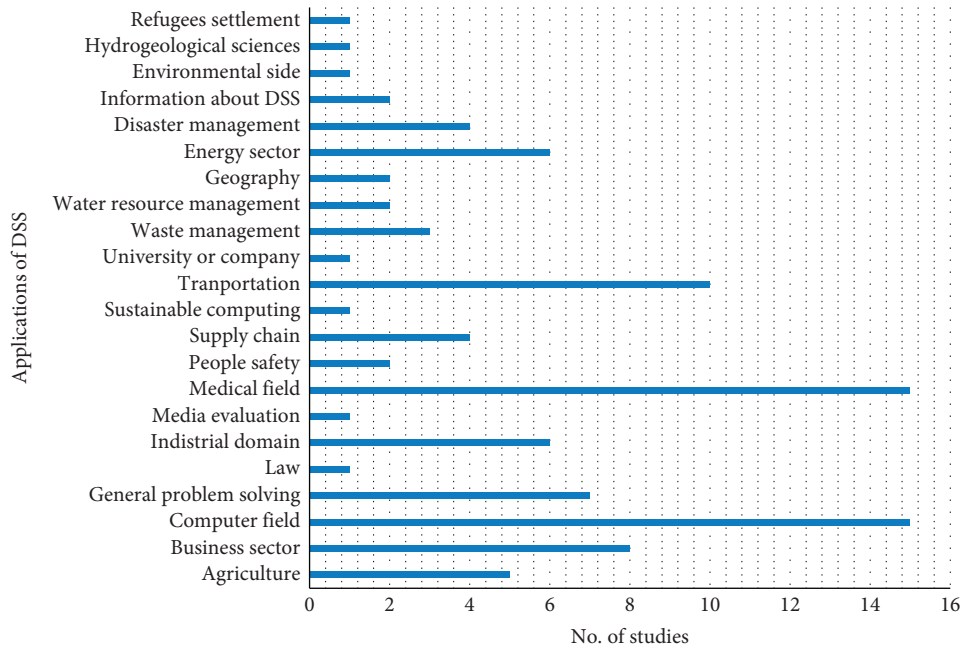


FIGURE 1: Application of the DSS and the studies published.

the evolution of the digital market. Different applications of DSS exist in various areas of research. Figure 1 depicts applications of DSS with studies in different areas.

The proposed study is endeavoured to identify the areas of DSS from diverse perspectives including the type of publications, year of publication, title of publication, and so on. For the initial search process, the query (“multi-criteria”) AND (“decision support system” OR “DSS”) AND (“Industrial Internet of Things” OR “IIoT”) was searched in the most popular libraries including ScienceDirect, Springer, IEEE, and ACM. Details of each library are given below. For ScienceDirect, the following information was gathered. Figure 2 represents the year-wise paper distributions with the amount of papers published.

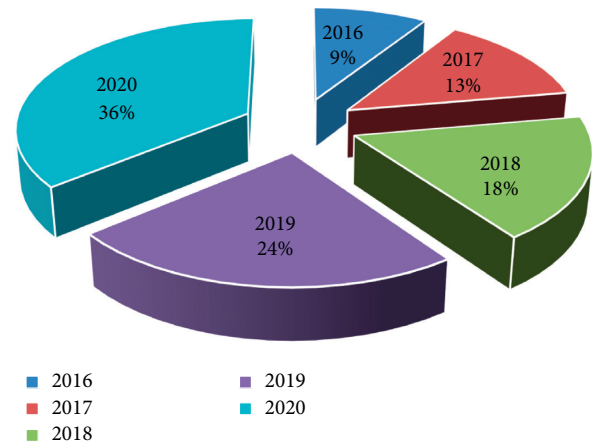


FIGURE 2: Years along with the total number of publications.

Figure 3 depicts publications titles with the total number of publications in the given library. From the figure, it is clear that the higher number of publications were done in the journal of Cleaner Production than in the journal of Future Generation Computer Systems followed by others.

Figure 4 presents the subject areas with the total number of publications in given library.

Figure 5 presents article types with the total number of articles. The figure reveals that the higher number of publications is research articles.

Guo et al. [42] proposed an intelligent DSS on ground of technology of data mining-applied enterprises for establishing of IoT-based smart DSS for the industry of manufacturing. The system supports the decision makers in early decision-making. The experimental results of the study show that the proposed technique of mining technology can analyse the data from various perspective of classifying, clustering, and modelling huge volume of data and identify the correlation between data. Mashal et al. [43] proposed a hybrid approach of multicriteria decision-making which relay on the analytic hierarchy process and simple additive

weight methods. The results of the study show that the application criterion is significantly important. The reliability, privacy, and availability were considered as the most crucial criteria of the applications of IoT.

After searching the library of ScienceDirect, the library of IEEE was searched in order to get significant information. Kashef et al. [44] proposed a tool of decision support for finding the cost optimum virtual machine settlement on multiclouds. The tool supports the model of cost estimation and optimization algorithm. Various simulations were done for examining the results achieved by the working of tool. Wan and Liao [45] proposed a coupling information system on ground of IoT and DSS exploring the research on the complete framework and design level of integration system. The authors [46] attempted for looking at the requirement of secure IIoT ecosystem in the standard of industry such as OpenFog consortium and industrial Internet consortium.

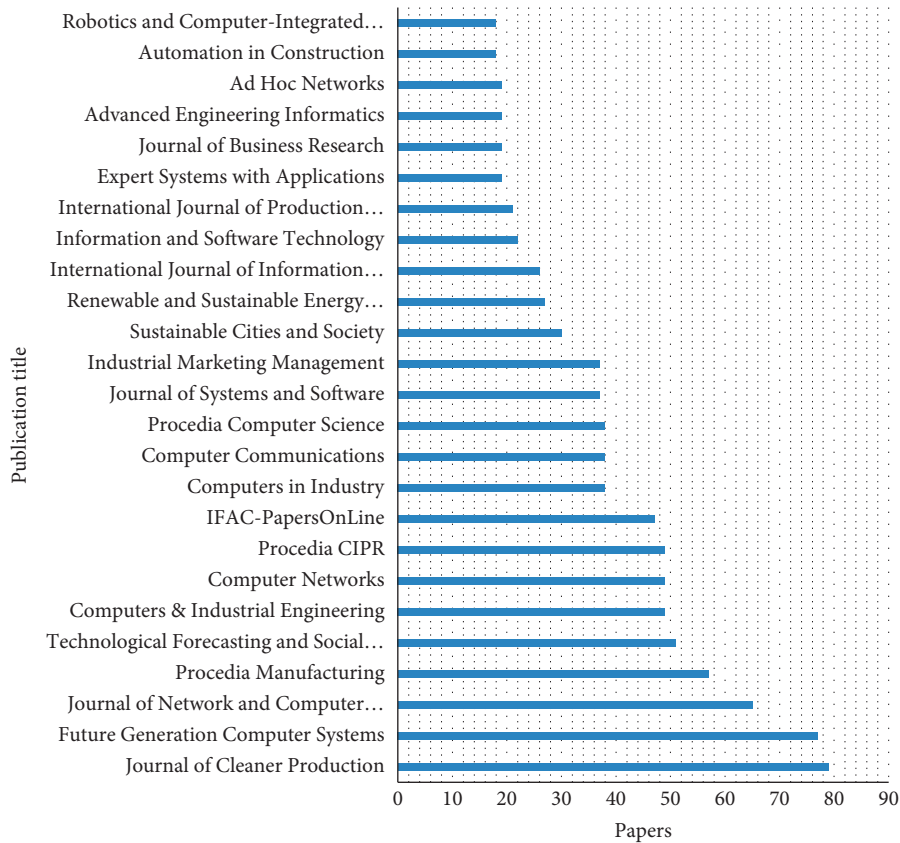


FIGURE 3: Title of publications with total number of publications.

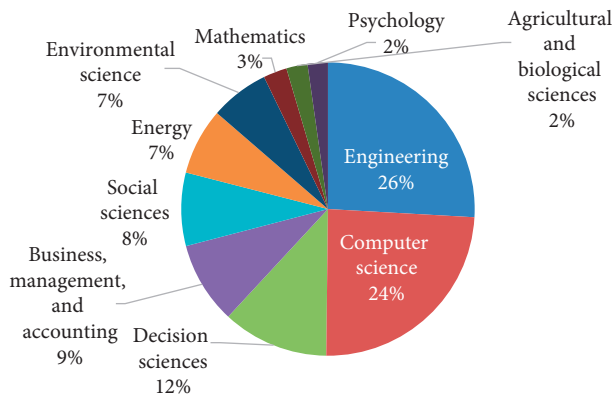


FIGURE 4: Subject area with articles.

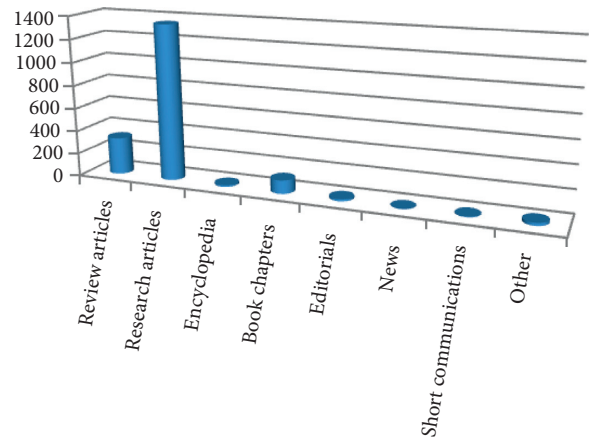


FIGURE 5: Type of articles with the total number of publications.

The study discussed the future directions of research for enhancing the privacy, security, and safety of the IIoT. Different types of information were obtained in the library. Figure 6 presents publication topics with the articles published in the IEEE library.

Figure 7 presents publications type with the total number of articles published in the given library. The figure depicts that the most number of paper types were conference followed by journal and then others.

The Springer library was part of the search process for obtaining the associated information. Sha et al. [47] proposed a system called IIoT-SIDefender which measures the

degree of security of sensitive information with the support of the AHP and TOPSIS. Silva and Jardim-Goncalves [48] proposed an approach for analysing a set of hardware choices according to the user requirements based on multicriteria and advice on appropriate solution of hardware for a particular situation. Figure 8 depicts the types of contents with papers published.

Figure 9 represents the discipline covered by given library with total number of papers.

The ACM library was considered as the relevant library for searching the associated information. Most of the IIoT

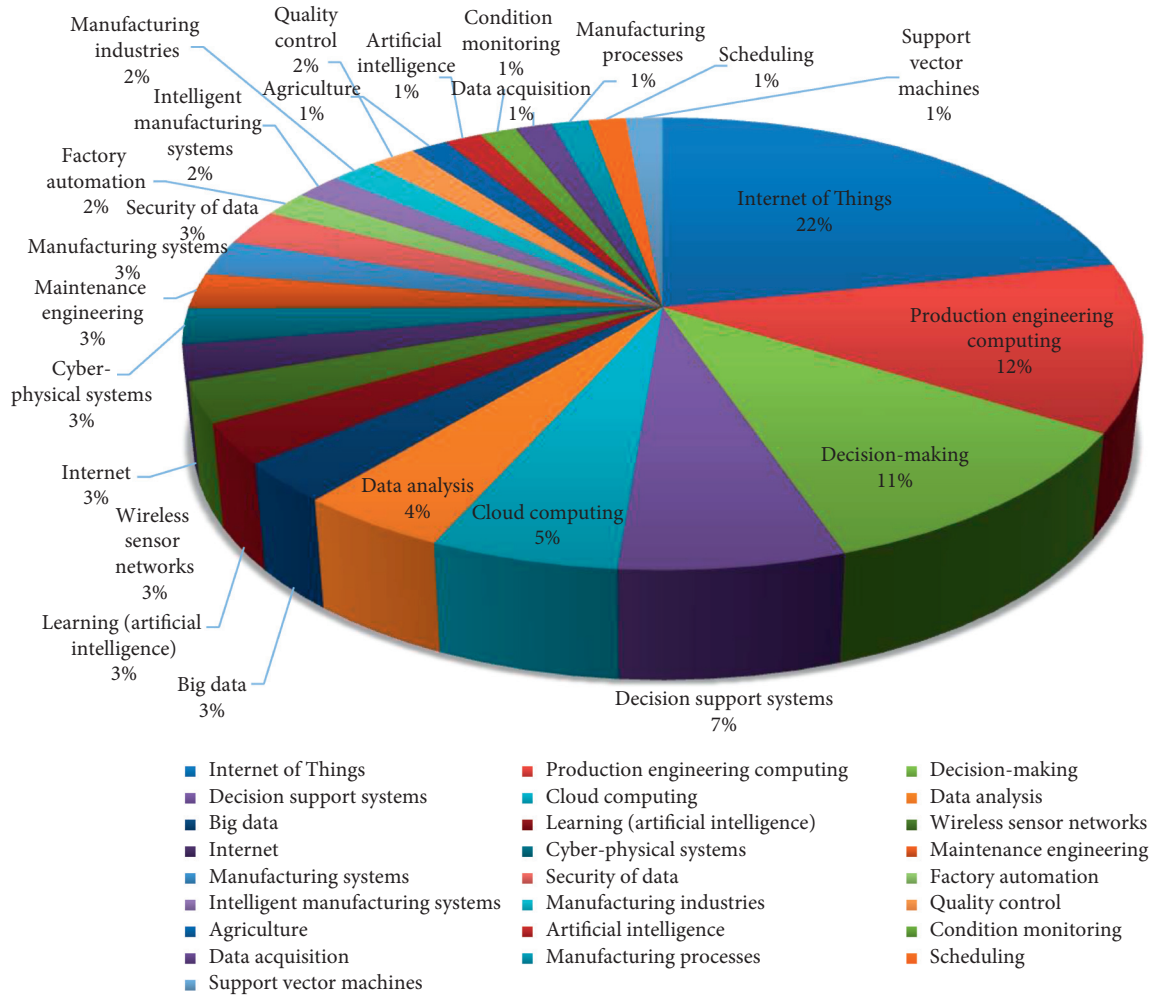


FIGURE 6: Publication topics along with total number of articles.

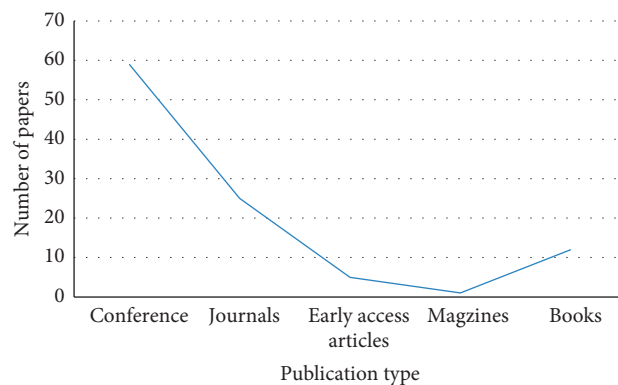


FIGURE 7: Article type with total number of publications.

based on DSS approaches were considered as part of this library. The study proposed an intelligent approach for facilitating the daily life with the technology of IoT. The ecosystem based on the application of particular network protocol for facilitating the transporting operation [49]. An approach was proposed based on multisensing integrating diagnosis system for real-time and accurate monitoring of large-scale machinery. The approach endeavours capturing

and modelling the temporal and spatial structure in sequential data and uses the mode-effective prediction of machinery [50]. Different types of information were gathered from the library of ACM. Figure 10 represents the total number of publications in given years with the total number of papers in the ACM library.

Figure 11 depicts the content types with the papers published.

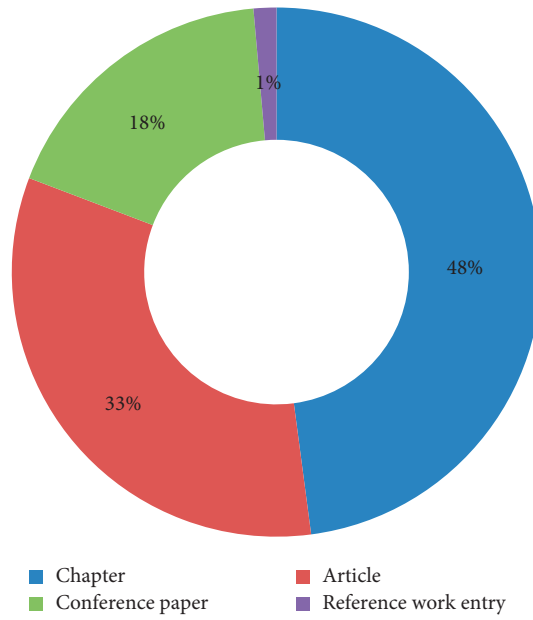


FIGURE 8: Content type with papers.

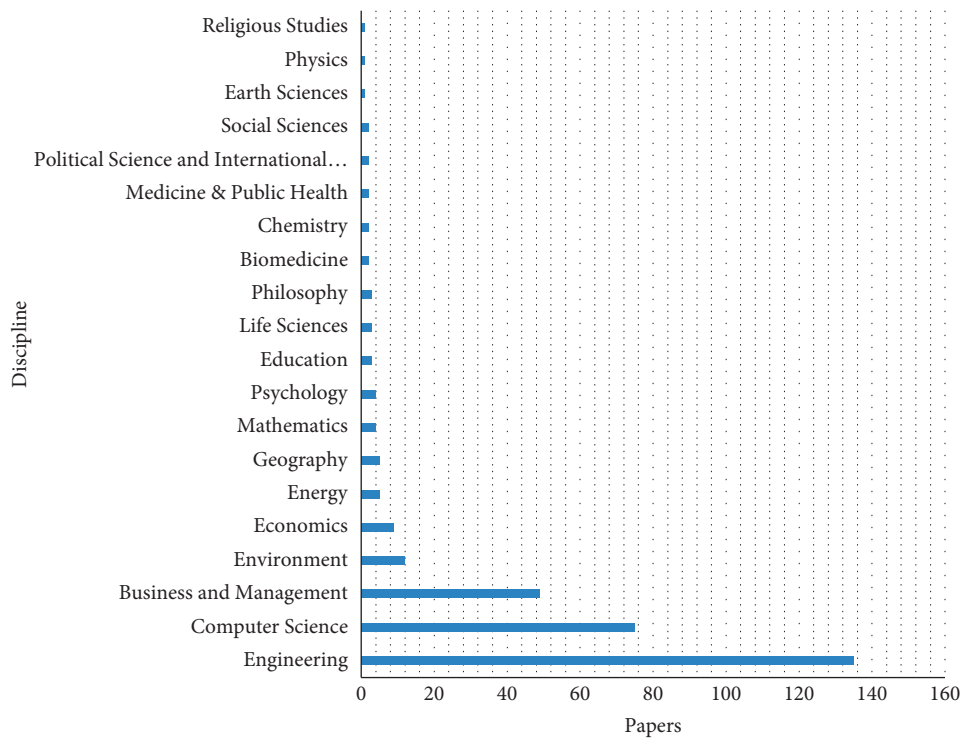


FIGURE 9: Discipline covered with articles.

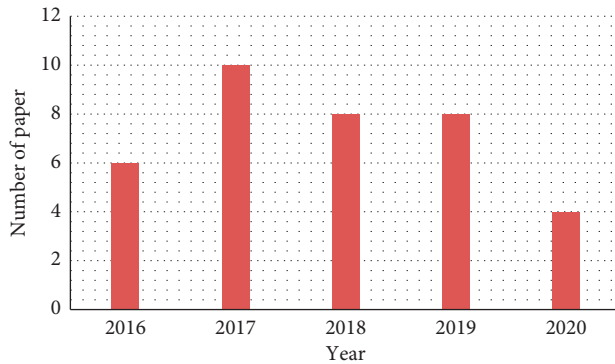


FIGURE 10: Publications based on the year.

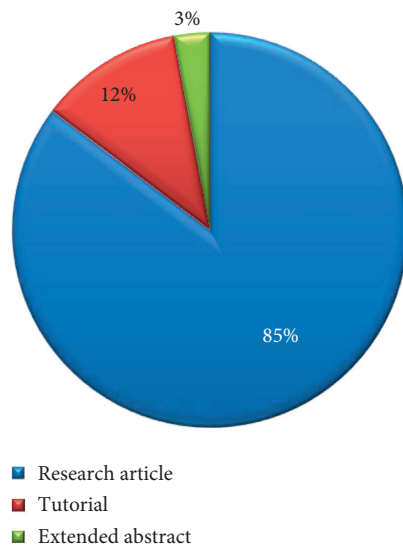


FIGURE 11: Content types along with the number of papers.

4. Conclusion

DSSs are playing a significant role in every field of life. With innovation in technology and information communications, decision-making on right and appropriate time is a challenging issue for the industrial Internet of things. Taking decision on the right and appropriate time can ultimately lead the industry into success. Different libraries were searched in order to identify the applications and domain areas of DSS in various fields of the IIoT. Based on the search process, numerous studies were identified associated with the DSS based on multicriteria in the area of IIoT. The study has exploited the power of DSS as an efficient alternative for solving complex problems of the IIoT based on various criteria for source code transformation. The study has analysed the searched papers from different perspectives of the contributions achieved by researchers in the field. The presented study provides beneficial understanding to the readers and experts of the domain to know the current status of research in order to provide more intelligent and effective solutions to cope with more complex decision-making issues in the field of IIoT.

Data Availability

No data are available.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This work was sponsored in part by Youth Program of National Natural Science Foundation of China (51804248) and Special Project of Education Department of Shaanxi Province, China (14JK1457).

References

- [1] M. Jemmali, M. Alharbi, and L. K. B. Melhim, "Intelligent decision-making algorithm for supplier evaluation based on multi-criteria preferences," in *Proceedings of the 2018 1st International Conference on Computer Applications & Information Security (ICCAIS)*, pp. 1–5, IEEE, Riyadh, Saudi Arabia, April 2018.
- [2] I. Aouadni and A. Rebai, "Decision support system based on genetic algorithm and multi-criteria satisfaction analysis (MUSA) method for measuring job satisfaction," *Annals of Operations Research*, vol. 256, no. 1, pp. 3–20, 2017.
- [3] S. Safdar, S. Zafar, N. Zafar, and N. F. Khan, "Machine learning based decision support systems (DSS) for heart disease diagnosis: a review," *Artificial Intelligence Review*, vol. 50, no. 4, pp. 597–623, 2018.
- [4] I. Petkovics, J. Simon, A. Petkovics, and Z. Covic, "Selection of unmanned aerial vehicle for precision agriculture with multi-criteria decision making algorithm," in *Proceedings of the 2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY)*, pp. 000151–000156, IEEE, Subotica, Serbia, September 2017.
- [5] C. Fleig, D. Augenstein, and A. Maedche, "Designing a process mining-enabled decision support system for business process standardization in ERP implementation projects," *Business Process Management Forum*, Springer, Cham, Switzerland, pp. 228–244, 2018.
- [6] A. Schwenk-Ferrero and A. Andrianov, "Nuclear waste management decision-making support with MCDA," *Science and Technology of Nuclear Installations*, vol. 2017, Article ID 9029406, 20 pages, 2017.
- [7] T. Ahmad, Y. Ma, M. Yahya, B. Ahmad, and S. Nazir, "Object detection through modified YOLO neural network," *Scientific Programming*, vol. 2020, Article ID 8403262, 10 pages, 2020.
- [8] Hanif-Ur-Rahman, H. K. Bamma, S. Nazir, S. Shahzad, and T. Hodosi, "A sourcing decision model for application maintenance services," in *Proceedings of the 3rd International Conference on Science in Information Technology (ICSITech)*, October 2017.
- [9] A. Khan, L. Jian Ping, H. Amin ul et al., "Partial observer decision process model for crane-robot action," *Scientific Programming*, vol. 2020, Article ID 6349342, 14 pages, 2020.
- [10] J. Li, A. Ullah, L. Jun et al., "Attributes based decision making for selection of requirements elicitation techniques using the analytic network process," *Mathematical Problems in Engineering*, vol. 2020, Article ID 2156023, 13 pages, 2020.
- [11] S. Nazir, S. Ali, M. Yang, and Q. Xu, "Deep learning algorithms and multi-criteria decision making used in big data: a

- systematic literature review,” *Complexity*, vol. 2020, Article ID 2836064, 18 pages, 2020.
- [12] S. Nazir, S. Shahzad, S. Mahfooz, and M. N. Jan, “Fuzzy logic based decision support system for component security evaluation,” *International Arab Journal of Information and Technology*, vol. 15, no. 2, pp. 1–9, 2015.
- [13] S. Nazir, S. Shahzad, A. Ullah, and A. Hussain, “Identification and analysis of project attributes affecting the decision of requirement elicitation technique,” in *Proceedings of the National Graduate Conference*, Islamabad, Pakistan, March 2017.
- [14] H. U. Rehman, M. Khan, Palwasha, H. U. Khan, and S. Nazir, “Analyzing factors that influence offshore outsourcing decision of application maintenance,” *IEEE Access*, vol. 8, 2020.
- [15] J. Zhang, S. Nazir, A. Huang, and A. Alharbi, “Multicriteria decision and machine learning algorithms for component security evaluation: library-based overview,” *Security and Communication Networks*, vol. 2020, Article ID 8886877, 14 pages, 2020.
- [16] L. S. R. Supriadi and L. Sui Pheng, “Knowledge based decision support system (KBDSS),” in *Business Continuity Management in Construction*, pp. 155–174, Springer, Singapore, 2018.
- [17] A. Alaeddini and K. G. Murty, “DSS (decision support system) for allocating appointment times to calling patients at a medical facility,” in *Case Studies in Operations Research: Applications of Optimal Decision Making*, K. G. Murty, Ed., Springer, New York, NY, USA, pp. 83–109, 2015.
- [18] M. Li, S. Nazir, H. U. Khan, S. Shahzad, and R. Amin, “Modelling features-based birthmarks for security of end-to-end communication system,” *Security and Communication Networks*, vol. 2020, Article ID 8852124, 9 pages, 2020.
- [19] S. Nazir, S. Shahzad, R. B. Atan, and H. Farman, “Estimation of software features based birthmark,” *Cluster Computing—The Journal of Networks Software Tools and Applications*, vol. 21, no. 1, pp. 1–14, 2017.
- [20] S. Nazir, S. Shahzad, S. A. Khan, N. Binti Alias, and S. Anwar, “A novel rules based approach for estimating software birthmark,” *The Scientific World Journal*, vol. 2015, Article ID 1579390, 8 pages, 2015.
- [21] S. Nazir, S. Shahzad, and N. Mukhtar, “Software birthmark design and estimation: a systematic literature review,” *Arabian Journal for Science and Engineering*, vol. 44, no. 4, p. 3905, 2019.
- [22] S. Nazir, S. Shahzad, R. Wirza et al., “Birthmark based identification of software piracy using Haar wavelet,” *Mathematics and Computers in Simulation*, vol. 166, pp. 144–154, 2019.
- [23] W. Liu, G. Huang, A. Zheng, and J. Liu, “Research on the optimization of IIoT data processing latency,” *Computer Communications*, vol. 151, pp. 290–298, 2020.
- [24] R. Sahal, J. G. Breslin, and M. I. Ali, “Big data and stream processing platforms for Industry 4.0 requirements mapping for a predictive maintenance use case,” *Journal of Manufacturing Systems*, vol. 54, pp. 138–151, 2020.
- [25] W. Z. Khan, M. H. Rehman, H. M. Zangoti, M. K. Afzal, N. Armi, and K. Salah, “Industrial internet of things: recent advances, enabling technologies and open challenges,” *Computers & Electrical Engineering*, vol. 81, p. 106522, 2020.
- [26] N. Gulati and P. D. Kaur, “Towards socially enabled internet of industrial things: architecture, semantic model and relationship management,” *Ad Hoc Networks*, vol. 91, p. 101869, 2019.
- [27] G. Aceto, V. Persico, and A. Pescapé, “Industry 4.0 and health: internet of things, big data, and cloud computing for healthcare 4.0,” *Journal of Industrial Information Integration*, vol. 18, p. 100129, 2020.
- [28] M. H. Rehman, I. Yaqoob, K. Salah, M. Imran, P. P. Jayaraman, and C. Perera, “The role of big data analytics in industrial internet of things,” *Future Generation Computer Systems*, vol. 99, pp. 247–259, 2019.
- [29] J. Ordieres-Meré, J. Villalba-Díez, and X. Zheng, “Challenges and opportunities for publishing IIoT data in manufacturing as a service business,” *Procedia Manufacturing*, vol. 39, pp. 185–193, 2019.
- [30] B. Diene, J. J. P. C. Rodrigues, O. Diallo, E. L. H. M. Ndoye, and V. V. Korotaev, “Data management techniques for internet of things,” *Mechanical Systems and Signal Processing*, vol. 138, p. 106564, 2020.
- [31] K. Alexopoulos, S. Koukas, N. Boli, and D. Mourtzis, “Architecture and development of an industrial internet of things framework for realizing services in industrial product service systems,” *Procedia CIRP*, vol. 72, pp. 880–885, 2018.
- [32] S. H. Almotiri, M. A. Khan, and M. A. Alghamdi, “Mobile health (m-health) system in the context of IoT,” in *Proceeding of the 2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, pp. 39–42, IEEE, Vienna, Austria, August 2016.
- [33] M. Younan, E. H. Houssein, M. Elhoseny, and A. A. Ali, “Challenges and recommended technologies for the industrial internet of things: a comprehensive review,” *Measurement*, vol. 151, p. 107198, 2020.
- [34] Y. Karaca, M. Moonis, Y.-D. Zhang, and C. Gezgez, “Mobile cloud computing based stroke healthcare system,” *International Journal of Information Management*, vol. 45, pp. 250–261, 2019.
- [35] M. Ge, H. Bangui, and B. Buhnova, “Big data for internet of things: a survey,” *Future Generation Computer Systems*, vol. 87, pp. 601–614, 2018.
- [36] V. Souza, R. Cruz, W. Silva, S. Lins, and V. Lucena, “A digital twin architecture based on the industrial internet of things technologies,” in *Proceedings of the 2019 IEEE International Conference on Consumer Electronics (ICCE)*, pp. 1–2, Las Vegas, NV, USA, January 2019.
- [37] D. Jiang, “The construction of smart city information system based on the Internet of Things and cloud computing,” *Computer Communications*, vol. 150, pp. 158–166, 2020.
- [38] L. Urquhart and D. McAuley, “Avoiding the internet of insecure industrial things,” *Computer Law & Security Review*, vol. 34, no. 3, pp. 450–466, 2018.
- [39] M. Humayun, N. Z. Jhanjhi, A. Alsayat, and V. Ponnusamy, “Internet of things and ransomware: evolution, mitigation and prevention,” *Egyptian Informatics Journal*, In press.
- [40] M. Dachyar, T. Y. M. Zagloel, and L. R. Saragih, “Knowledge growth and development: internet of things (IoT) research, 2006–2018,” *Heliyon*, vol. 5, no. 8, p. e02264, 2019.
- [41] S. Gierej, “The framework of business model in the context of industrial internet of things,” *Procedia Engineering*, vol. 182, pp. 206–212, 2017.
- [42] Y. Guo, N. Wang, Z.-Y. Xu, and K. Wu, “The internet of things-based decision support system for information processing in intelligent manufacturing using data mining technology,” *Mechanical Systems and Signal Processing*, vol. 142, p. 106630, 2020.
- [43] I. Mashal, O. Alsaryrah, T.-Y. Chung, and F.-C. Yuan, “A multi-criteria analysis for an internet of things application recommendation system,” *Technology in Society*, vol. 60, p. 101216, 2020.

- [44] M. M. Kashef, H. Yoon, M. Keshavarz, and J. Hwang, "Decision support tool for IoT service providers for utilization of multi clouds," in *Proceedings of the 2016 18th International Conference on Advanced Communication Technology (ICACT)*, pp. 91–96, Pyeongchang, South Korea, February 2016.
- [45] H. Wan and L. Liao, "A coupling system design based on the internet of things and intelligent decision support system in industrial enterprises," in *Proceedings of the 2016 3rd International Conference on Information Science and Control Engineering (ICISCE)*, pp. 753–757, Beijing, China, July 2016.
- [46] T. Gebremichael, L. P. I. Ledwaba, M. H. Eldefrawy et al., "Security and privacy in the industrial internet of things: current standards and future challenges," *IEEE Access*, vol. 8, pp. 152351–152366, 2020.
- [47] L. Sha, F. Xiao, W. Chen, and J. Sun, "IIoT-SIDefender: detecting and defense against the sensitive information leakage in industry IoT," *World Wide Web*, vol. 21, no. 1, pp. 59–88, 2018.
- [48] E. M. Silva and R. Jardim-Goncalves, "Multi-criteria analysis and decision methodology for the selection of internet-of-things hardware platforms," in *Proceedings of the Doctoral Conference on Computing, Electrical and Industrial Systems*, pp. 111–121, Springer, Costa de Caparica, Portugal, May 2017.
- [49] F. Z. Chafi and Y. Fakhri, "The integration of multi agent system within the internet of things: the use of SigFox shield as a network," in *Proceedings of the 3rd International Conference on Smart City Applications*, pp. 1–8, Tetouan, Morocco, October 2018.
- [50] H. Liu, "A multi-sensing collaborative diagnosis system for the reliability of industrial IoT," in *Proceedings of the 2019 International Conference on Embedded Wireless Systems and Networks*, pp. 391–400, Beijing, China, 2019.

Research Article

Software Piracy Awareness, Policy, and User Perspective in Educational Institutions

Zitian Liao ^{1,2} Shah Nazir ³ Anwar Hussain,³ Habib Ullah Khan ⁴
and Muhammad Shafiq⁵

¹University of Sydney, School of Architecture Design & Planning, New South Wales 2006, Sydney, UK

²School of Electronic Engineering, Xidian University, Xi'an 710126, China

³Department of Computer Science, University of Swabi, Swabi, Pakistan

⁴Department of Accounting & Information Systems College of Business & Economics Qatar, Qatar University, Doha, Qatar

⁵Cyberspace Institute of Advance Technology, Guangzhou University, Guangzhou, China

Correspondence should be addressed to Zitian Liao; zitianliao@sina.com

Received 10 October 2020; Revised 17 November 2020; Accepted 27 November 2020; Published 11 December 2020

Academic Editor: Sikandar Ali

Copyright © 2020 Zitian Liao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Software theft and piracy are rapidly ever-increasing problems of the present-day software industry. Software piracy is the illegal copy and use of software in a way other than that is officially documented by exclusive rights of the developer in the form of an individual or organization as described in the relevant sale agreement (license). Owing to the evolution in software development and Internet, software piracy has become a main concern for many software companies. Software companies are confronted with extremely high losses due to the piracy of software. Pirates achieve a lot of money by doing business with pirated software. General end-users of the software are not aware of this serious crime and of the legal consequences of breaking the law. Even most of the time, end-users and consumers think that it is none of their concern and not an important issue for them. Although, in reality, if an organization is working with pirated software, there is a risk of failure of the software, and it might put their organization at risk as pirated software does not receive any support from the development organization. This ultimately puts the consumer organization in huge financial loss. Due to these reasons, software piracy has turned out to be a major concern, more emergent due to the extravagant development of the software industry and the availability of software(s) on the Internet. In this paper, we analyzed and identified the ratio of software piracy, awareness regarding piracy, and the policy of the licensed software provided. Based on the results of the study, some suggestions are proposed by which the level of piracy can be reduced.

1. Introduction

Software piracy is the illegal copying, installation, use, distribution, or sale of software in any way other than that is expressed in the license agreement. The software industry is facing huge financial losses due to the piracy of software. Piracy of software is performed by end-users as well as by the dealers. It causes serious problems that hinder the success of the software industry nationwide and globally. The pirates gain effortless benefits from the sale of pirated software and this ultimately affects the business of the software industry. Piracy of software is the legal consequences of breaking the law. Piracy is performed in different ways, such as hard-disk

loading, soft lifting, counterfeit goods, rental software, and bulletin board piracy [1–4]. The original licensed software offers a number of high valued benefits to the customers and users, like upgrades are available, assurance of quality and reliability, technical support, manuals or documentation, no exposure of your network to security breaches, while the pirated software fails to do so [5]. An organization with the use of pirated software might put them at a huge financial loss, as they are using pirated software that does not provide the mentioned benefits.

Researchers have been attempting to develop techniques to easily detect, prevent, and identify piracy performed in the software [6–10]. Still, there is a shortage of knowledge about

the intricate details of piracy and methodologies, which could aid to notice software piracy in a resourceful way. Along with this, there is a need to create social awareness about the piracy of software and to create a culture of being honest by using only original and legal software. Piracy of software is an observable fact that causes problems for all stakeholders involved, from owners to developers, distributors, vendors, to end-users as well. Organizations and end-users are to be disheartened from consuming pirated software which is not only the theft of rights of the owners and developers of the software, but it might also put them in serious difficulty and high losses. With this kind of social awareness, along with the technical protection against piracy, there will be a gradual decrease in the use of pirated software which will ultimately result in bringing lost profits back to the software industry and the industries will work in a better way.

The cognition process behind software piracy and one of the main reasons for piracy is the psychological factor for not considering it as a crime, which is ultimately a threat to the software industry. So, in order to reduce the piracy of software, it is more important to address that, what are the cognitive reasons or psychological factors behind it. In order to tackle these limitations, the proposed study identifies the main factors of piracy, and then based on these factors, some suggestions are proposed.

The contribution of this paper is to find the existing level of software piracy performed by customers and users, awareness of the piracy, policy of the licensed software, and user perspectives in educational institutions of Pakistan. Furthermore, the main contribution of this paper is given below:

- (i) To identify the level of existing software piracy in educational institutions done by users
- (ii) To quantify the existing awareness regarding the use of illegal software
- (iii) To find the level of awareness of the policy regarding original and licensed software
- (iv) To identify the reasons behind software piracy in academia
- (v) To propose suggestions/solution for how to reduce software piracy based on the above discussion

Based on the experimental results, some suggestions are proposed by which the level of piracy can be reduced. These suggestions include “suitable methods of payment for software purchasing,” “availability of Internet in academic institutions,” “conducting seminars on software piracy in academia,” “awareness of piracy,” “HEC visits for ensuring the implementation of their policies in the academia,” “decreasing software costs and licenses prices,” and “implementation of software policy in academia.”

The remainder of the paper is organized as follows. Section 2 represents related work to software piracy issues and their possible solutions. Section 3 shows the details of the proposed methodology carried out. Section 4 gives results and discussions of the proposed methodology. Section 5 mentions the conclusions of this research.

2. Related Work

The software industry and community of researchers have been attempting to develop techniques that can easily detect, prevent, and identify piracy in software. Several diverse techniques are available for the same, but still, there is a lack of knowledge about piracy and the methodologies used for piracy to detect software piracy in an efficient way. Apart from this, there is a need to create awareness for avoiding software piracy and to develop a tradition of being honest by using only original, legal, and licensed software.

The existing methodologies provide enough details for piracy detection and avoidance. Peukert et al. [11] have evaluated the heterogeneous effects of online copyright enforcement. Robertson et al. [12] analyzed the patterns of software piracy for the 20 nations of Latin America. Gan and Koh [2] used a survey technique at the three universities of Singapore for examining the perception of software piracy and to discover the mentioned factors. Mumtaz et al. [13] developed a methodology for piracy protection of secure electronic software distribution.

Mo et al. [14] investigated the opportunity and the setting for their revenue sharing by online content piracy monitoring for Internet service providers and content providers. They further investigated the ISP’s piracy monitoring cost level, the value of contents, and control provider access fee. Lowry et al. [15] conducted a meta-analysis of the literature and analyzed 257 studies with 126,622 participants for investigating the main constructs and covariates. Their meta-analysis results suggest four key sets of factors maximize predictions which are outcome expectations, social learning, self-efficacy, and moral disengagement. Kumar et al. [16] presented a secure split test with functional test capability to mitigate the counterfeits coming from untrusted foundries.

Huang et al. [17] presented a study that considers a single supplier who may sell pirated goods through two independent and different retails channels (traditional and digital). A Stackelberg game is utilized to determine the optimal gain sharing ratio and the equilibrium price for all channel members. Their study found that an increase in piracy would force retailers to compete in a smaller market and lead to a decrease in profits for members of the channel. Chang et al. [18] presented a study that examines the factor effects of software piracy at the country level. From their study, it was found that economic development, trade, education, freedom, regulatory protection, and computer penetration all drastically affect the level of software piracy within the country.

Rasch and Wenzel [19] worked on a two-sided market setting of the impact of software piracy, which includes software platforms that attract developers and users to maximize their profits. Banerjee [20] analyzed the impact of instantaneous increase in piracy and network externalities on research and development investment. Siponen et al. [21] developed a model that explains the effects of neutralization techniques on the intention of software piracy. The results showed that appeal to higher loyalties and condemn the

condemners highly predict the intention of software piracy. Andrés and Goel [22] examine the effect of software piracy on medium term growth using cross-country data from 2000 to 2007. Their findings suggest that piracy of software reduces economic growth over the medium term. Kariithi [23] describes the related work to music, film, and piracy of software around the globe, with the attention to data sources, research scope, and generic findings. The author finds that the absence of methodologies utilizing critical theory in this broad literature has constricted the world view of piracy.

Marti'nez-Sanchez [24] analyzed the government and incumbent role in preventing the pirate entry. The framework used a sequential duopoly model of vertical product differentiation with price competition. The results show that both the government and the incumbent have a major role in preventing pirate entry. Al-Rafee and Rouibah [25] reports experiments to prevent digital piracy in Arab and Middle Eastern countries. The experimental results showed that only the religion and awareness treatments contributed to turning down piracy. Nill and Shultz [26] provided an overview of international legal, systematic, and economic considerations and shared an analysis of the drivers of software piracy consumers. The authors discussed strategic considerations and a decision-making typology is introduced which helps legitimate companies to plan strategies in the face of widespread piracy. Peitz and Waelbroeck [27] provided a critical review of the theoretical literature which addresses the economic consequences of end-user copying.

Hamade [28] described the legal and political aspects of software piracy in general and specifically in the Arab world. Banerjee [29] used a framework to address the issue of public policy regarding anticommercial piracy. Bae and Choi [30] developed a model of software piracy to analyze the short-run effects of piracy on the usage of software and the long-run effects of development incentives. Fung and Lakhani [31] analyzed the potential end-user copyright violations linked with peer to peer file sharing and anti-piracy efforts. Png [32] concluded that the consultant and methodology change in Business Software Alliance in 2002-2003 had systematic effects on published piracy rates. The decrease trend rate of piracy falls from 2.0% to 1.1% points per year. The proposed research is an endeavor toward identifying the level of existing software piracy in educational institutions done by users, finding the level of how much information is there about awareness of software piracy, finding the level of awareness of the policy regarding original and licensed software, identifying the reasons behind software piracy in academia, and proposing suggestions/solution for how to reduce software piracy based on the above discussions.

3. Proposed Approach

The following sections discuss the proposed approach and experimental study.

3.1. Software Piracy. The piracy of software causes serious problems that hinder the success of the software industry in

the national and international markets. The comparison of original licensed software with pirated software shows what benefits the user gets. The original software offers a number of high valued benefits to the customers, including assurance of software quality, availability of upgrades, technical and manual documentation, and less bandwidth consumption. On the other hand, pirated software fails to do so. There might be a risk of failure of the system if an organization was using pirated software, and pirated software might put the organization at the risk of huge financial loss. Some software is available in the form of open-source. But this open-source software is mostly licensed and needs a proper license agreement. Pirates are doing piracy of such software, which ultimately gives loss to the owners [9].

3.2. Protocol of the Study and Experimental Setup. The proposed study was conducted to identify the impact of software piracy in educational institutions. The first step of the study is to find the current level of piracy, its awareness, and the reasons behind why people are doing piracy. This paper addresses the following research questions which are based on a study of the literature and market:

- (a) Is the software piracy rate high in academia?
- (b) Are people aware of the software piracy issue?
- (c) Why do people commit piracy and what are the main reasons behind this issue?
- (d) What could be the possible solutions to reduce software piracy in academic institutions?

The study has the following research hypotheses:

- (i) Null hypothesis = Ho: piracy rate is not high in educational institutions
- (ii) Alternative hypothesis = H1: piracy rate is high in educational institutions
- (iii) Null hypothesis = Ho: people do have much awareness of software piracy
- (iv) Alternative hypothesis = H2: people do not have much awareness of software piracy
- (v) Null hypothesis = Ho: academic institutions are fully utilizing the Higher Education Commission (HEC) software facilities for its employees
- (vi) Alternative hypothesis = H3: academic institutions are not fully utilizing HEC software facilities for their employees

Rejection of the null hypotheses will lead to the acceptance of our alternative hypotheses which will validate the need and relevance of the conducted study. Figure 1 shows the protocol followed in the proposed study.

In this context, a survey has been conducted through a questionnaire consisting of total of 38 questions related to software piracy. Questions are divided into five sections, which are shown in Table 1.

For conducting a survey of proposed research work, a questionnaire was designed and sent to the faculty members, students, and administrative staff of different universities of

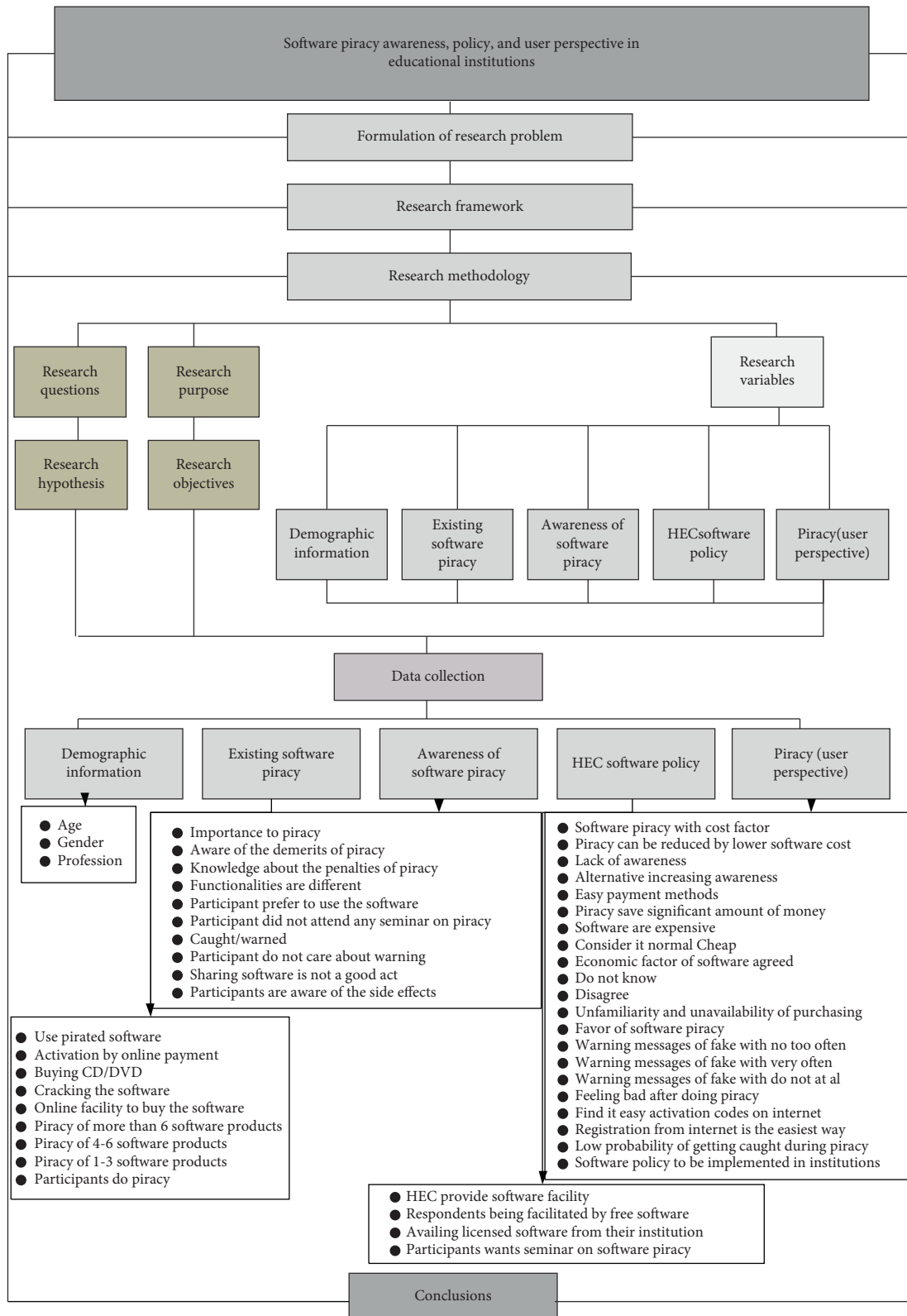


FIGURE 1: Protocol of the proposed study.

TABLE 1: Categories of questions.

S. no.	Section	No. of questions	Purpose
1	Demographic information	3	Information about age, gender, and profession
2	Existing level of software piracy	6	To identify what is the existing level of software piracy
3	Awareness of software piracy	12	To know how much people are aware of software piracy, its merits, and demerits, etc.
4	HEC software policy awareness and its availability	4	To identify the awareness of people about HEC policies for software facilities to academic institutions
5	Reasons behind software piracy	13	To know why people do software piracy

the country. A total of 110 responses were received. These responses were analyzed using SPSS software.

4. Results and Discussion

The questionnaire was sent to more than 500 hundred people including faculty members, students, and administrative staff of different universities. They were contacted through their official e-mail. A total of 110 responses were received from 37 universities in Pakistan. The data has been analyzed by means of the SPSS tool to determine whether the results had statistically significant differences. For the tests, we used a confidence interval of 90% and a significance level of 0.05. Null hypotheses H_0 became rejected when the p values were less than 0.05. Summary of the test for the existing level of piracy, piracy awareness, utilization of HEC software facilities, and reasons are provided in the following sections.

It is important to note that SPSS test summary tables are aimed to show whether the difference in the responses is significant or not. Null hypothesis terminology in Tables 2–5 refers to the different null hypotheses of the study represented as H_0 . Based on the responses received, the following subsections show the results achieved from the study conducted.

4.1. Existing Level of Piracy. Most of the people in universities are using pirated software. Statistics of the present study shows that 67.3% of people do use pirated software. The activation of the software is done by using fake (illegal) cracks and other activation methods. The fake key is used as an alternate for showing the software is original, while in the actual original software, it is allowed only to those users who purchased the license of the software. Using fake keys is the piracy of software is a serious crime. According to the statistics of the study, the activation by online payment is too low, which is 7.3%. Buying CD/DVD from the market is still less (38.2%) compared to activation by cracking the software (54.5%). The online payment facility in educational institutions shows that 81.8% of the participants do not have to buy the software online and pay for it, while only 18.2% have the facility to buy the software online. The rates of the total number of pirated software were identified to be much higher from the survey. According to the survey, 49.1% of each participant uses more than six pirated software. Other participants are not exempted from piracy but differ only in less number of pirated software, and their ratio is 18.2% for

(4–6) number of pirated software and 32.7% for (1–3) number of pirated software. The study shows that the existing piracy is too high and the majority of people use pirated software. About 80% of participants think that people do piracy. Figure 2 shows the details of a different aspect of the existing level of piracy.

The summary statistics for the existing level of piracy can be seen in Table 2. The null hypothesis has been rejected for each variable aimed at identifying the existing level of piracy which shows that the difference among the responses is significant. By summarizing the results, we can say that the Null hypothesis “piracy rate is not high in educational institutions” is rejected.

Figure 3 shows the representation of different groups of variables in the area from the current research perspectives. These variables include the use of pirated software, activation by online payment, buying CD/DVD, cracking the software, online facility to buy the software, piracy of more than 6 software products, piracy of 4–6 pieces of software, piracy of 1–3 software, participants doing piracy, importance of piracy, aware of the demerits of piracy, knowledge about the penalties of piracy, different functionalities, participant preferring to use the software, participants not attending any seminar on piracy, caught/warned, participants not caring about warning, sharing software not considered a good act, and participants being aware of the side effects. These variables were taken as important considerations of the proposed research. The relevant values of these variables are given in Figure 3.

4.2. Awareness about Software Piracy. The awareness of software piracy is analyzed through the questionnaire. Piracy of software is an important issue for participants to be stopped. The survey statistics show that only 1% does not give importance to piracy. The disadvantages and ethics of piracy show that most of the participants, 70.9%, are aware of the demerits of piracy, while the rest of the participants are not aware of the disadvantages of piracy. They agree that piracy is ethically wrong to do. The people know about the disadvantages of piracy but not their penalties for doing piracy. Only 34.5% have knowledge about the penalties of piracy, while the rest are unaware of it. The pirated software is functionally different from the licensed software. Among all participants, 58.2% agree that their functionalities are different, while 29.1% do not notice that they are different at all. The selection criteria for software products shows that

TABLE 2: Different aspects of the existing level of piracy.

S. no.	Null hypothesis	Test	Sig.	Decision
1	The categories defined by soft_use = Pirated and licensed occur with probabilities 0.5 and 0.5	One-sample binomial test	0.000	Reject the null hypothesis
2	The categories of friend_use occur with equal probabilities	One-sample chi-square test	0.000	Reject the null hypothesis
3	The categories of soft_active occur with equal probabilities	One-sample chi-square test	0.000	Reject the null hypothesis
4	The categories defined by online_pay = No and yes occur with probabilities 0.5 and 0.5	One-sample binomial test	0.000	Reject the null hypothesis
5	The categories of Pirated_soft_use occur with equal probabilities	One-sample chi-square test	0.000	Reject the null hypothesis
6	The categories of people_piracy occur with equal probabilities	One-sample chi-square test	0.000	Reject the null hypothesis

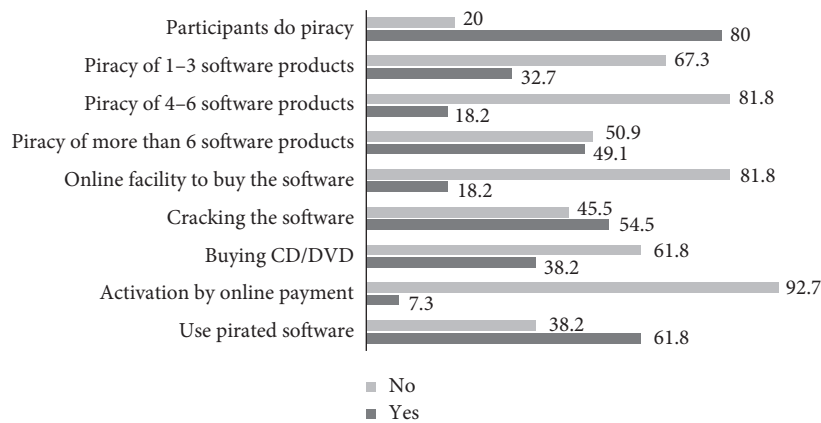


FIGURE 2: Different aspects of the existing level of piracy.

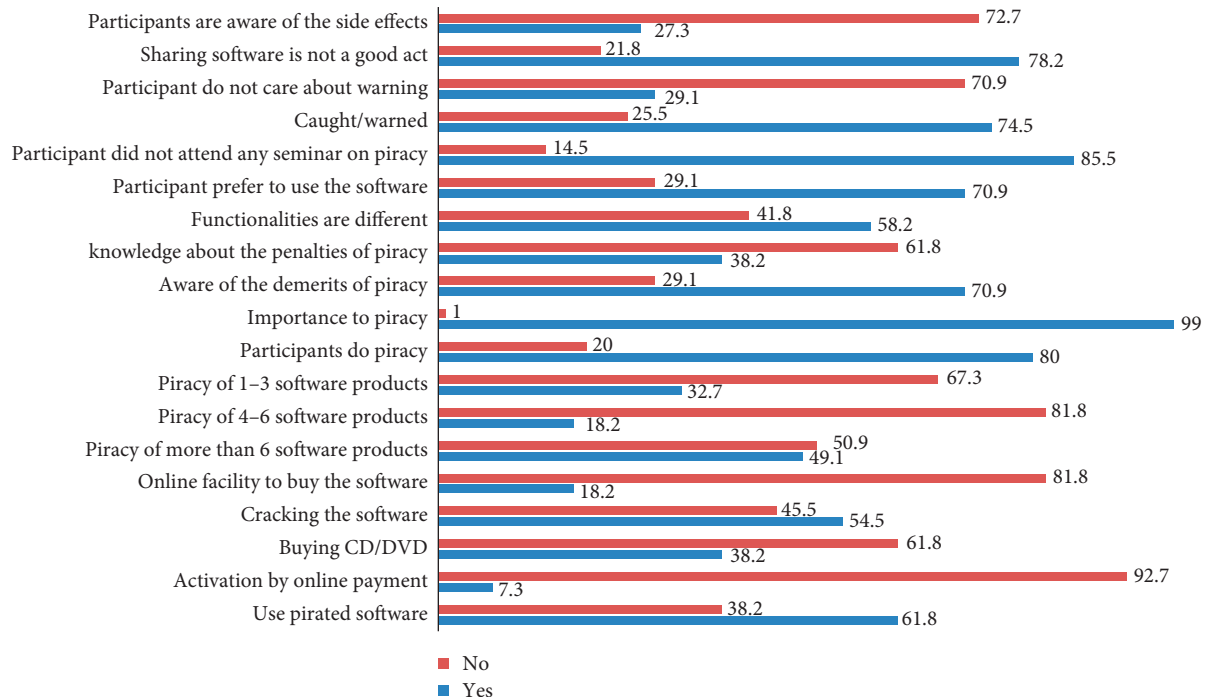


FIGURE 3: Tree representation of different groups of variables in the area.

70.9% of the participants prefer to use the software that provides rich number of features and additional features come with more expensive license cost. The responses regarding the conduction of awareness seminars show that 85.5% of participants did not attend any seminar on piracy and its related issues. The users have been caught/warned while using pirated software. The statistics of this study for caught/warned participants show that most of them have been caught at least 1–3 times (74.5%) and 16.4% of participants have a much higher rate of warning (more than 6 times). People do not feel embarrassed while using pirated software. But the difference between the frequency of the response is not significant enough. People share their pirated software with other friends and colleagues, but still, they agree that this is not a good act, 78.2%. Participants are not aware of all side effects of software piracy and only a few of the participants have awareness, 27.3%, while 72.7% of participants are not aware at all. Figure 4 shows the awareness of software piracy.

The statistics of awareness about piracy are shown in Table 3. The null hypothesis is rejected for all the variables except feeling shame while using pirated software. From the results, it can be concluded that the null hypothesis “people do have awareness about software piracy” is not being fully retained; the results show that people have awareness about software piracy on the basic level. The awareness is lacking penalties that can be given to those who commit piracy of software. They do not know all the side effects of software piracy. They have not attended any seminars on software piracy and the most important thing they are involved in piracy because of only having basic knowledge.

4.3. HEC Software Policy Facility Utilization. The HEC provides the facility of the licensed software to use and have different plans for academic institutions. A total of 45.5% of the survey respondents know about this facility of HEC and 54.5% are not aware of it. The difference between their responses is not significant enough. So, it cannot be concluded that participants know about the HEC software facility. The institution’s role in software availability was studied. It has been noticed that people use their own software and the academic institutions do not provide them any free software facility. A total of 67.3% of respondents is not being facilitated by any kind of free software from their institution. Most of the academic institutions do not facilitate their faculty members, students, and other staff by purchasing licensed software. Only 29.1% of participants have been availing of licensed software from their institution. Participants do not have enough knowledge about software facilities that they can avail of and can be provided by their institutions. In this regard, almost all participants (96.4%) want to have a seminar on software piracy. Figure 5 shows the details of the HEC software facility policy utilization.

The statistics for utilization of HEC software facilities are shown in Table 4. HEC has software policies to provide software to academic institutions. From the statistics given in Table 4, the ratio of awareness about HEC software

facilities is not significant enough. Other measures for utilization of HEC software facilities by academic institutions are still the main point of concern. In Table 4, it can be seen that the null hypothesis is rejected for the similarity of the responses for variables. It shows that participants are not being facilitated by academic institutions and seminars are required to be conducted both for academic institutions, along with participants. So, the main null hypothesis “academic institutions are fully utilizing HEC software facilities to its employees” is being rejected, which means academic institutions are not fully utilizing HEC software facilities to its employees.

4.4. Software Piracy and Their Cause from User Perspective. High software cost is one of the reasons behind software piracy. From the survey statistics, about 70% of participants consider it because of the price, while another reason is lack of awareness that has statistics of 23.6%. Software piracy ratio can be reduced by lower software cost, as in survey its statistics are high (54.5%) as compared to other alternative increasing awareness (30.9%) and easy payment methods (14.5%). Piracy of software saves money and prices of paid software as most of the participants (72.7%) say that piracy saves a significant amount of money for them because software prices are high. From statistics, the software is expensive for 85.5% of the survey members, while others consider it normal (10.9%) or cheap (4.6%). The economic factor of software was studied and the study shows that economy is an incentive for purchasing pirated software. 52.7% of participants agreed, while others did not know (30.9%) or disagreed (16.4%).

Unfamiliarity and unavailability of online purchasing is a factor of the survey in which most of the people (67.8%) are not able to buy software directly from the Internet because they do not have an online buying facility. The participants were not in favor of software piracy. Only 32.7% of the participants are on the other side. The participants receive fake software registration notifications not too often (50.9%), while only 25.5% of participants receive notifications very often and 23.6% do not receive them at all. Receiving warnings and notifications also has a psychological bad effect on the participants. They do not like at all the warning they receive for doing piracy. A total of 61.8% of participants feel bad after this, while 29.1% do not care. Still, some of them feel happy to receive it.

Finding fake/pirated software activation codes and licenses on the Internet is easy and statistics do not differ significantly (54.5%). In contrast, 45.5% of participants find it easy to find activation codes on the Internet. The difference is not significant enough and we cannot say that activation codes can easily be found on the Internet based on the higher percentage of responses (54.5%). Also, registration of software from the Internet is the easiest way (70.9%) for people are compared to other alternatives. The low probability of getting caught during piracy is medium for 49.1% of participants, while 41.8% of the participants’ probability of being caught is low. Poor implementation of software policy has been studied and the execution of software policy needs

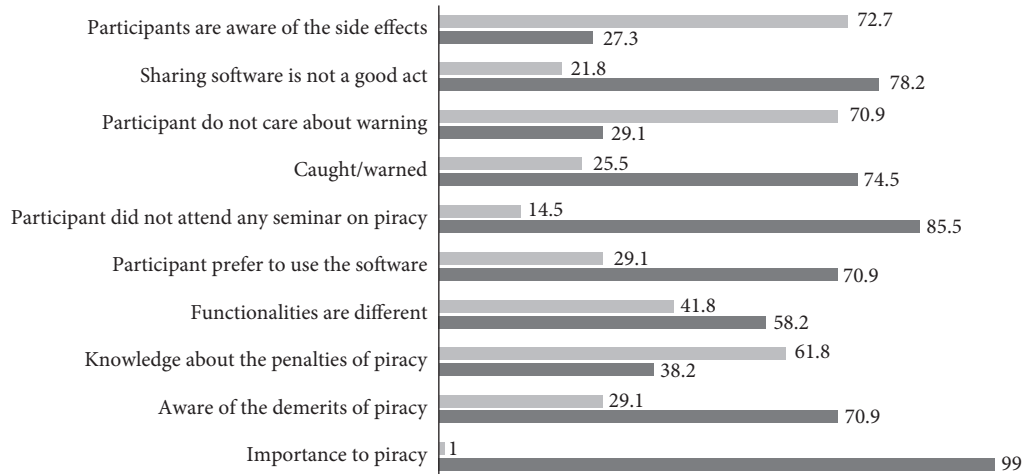


FIGURE 4: Awareness about software piracy.

TABLE 3: Awareness about software piracy.

S. no.	Null hypothesis	Test	Sig.	Decision
1	The categories defined by soft_use = Pirated and licensed occur with probabilities 0.5 and 0.5	One-sample binomial test	0.000	Reject the null hypothesis
2	The categories of friend_use occur with equal probabilities	One-sample chi-square test	0.000	Reject the null hypothesis
3	The categories of soft activate occur with equal probabilities	One-sample chi-square test	0.000	Reject the null hypothesis
4	The categories defined by online_pay = No and yes occur with probabilities 0.5 and 0.5	One-sample binomial test	0.000	Reject the null hypothesis
5	The categories of pirated_soft_use occur with equal probabilities	One-sample chi-square test	0.000	Reject the null hypothesis
6	The categories of function_diff occur with equal probabilities	One-sample chi-square test	0.000	Reject the null hypothesis
7	The categories defined by attend_seminar = No and yes occur with probabilities 0.5 and 0.5	One-sample binomial test	0.000	Reject the null hypothesis
8	The categories of caught_piracy occur with equal probabilities	One-sample chi-square test	0.000	Reject the null hypothesis
9	The categories of feel_shame occur with equal probabilities	One-sample chi-square test	0.000	Retain the null hypothesis
10	The categories defined by sharing_piracy = Agree and disagree occur with probabilities 0.5 and 0.5	One-sample binomial test	0.000	Reject the null hypothesis
11	The categories defined by side effect = No and yes occur with probabilities 0.5 and 0.5	One-sample binomial test	0.000	Reject the null hypothesis

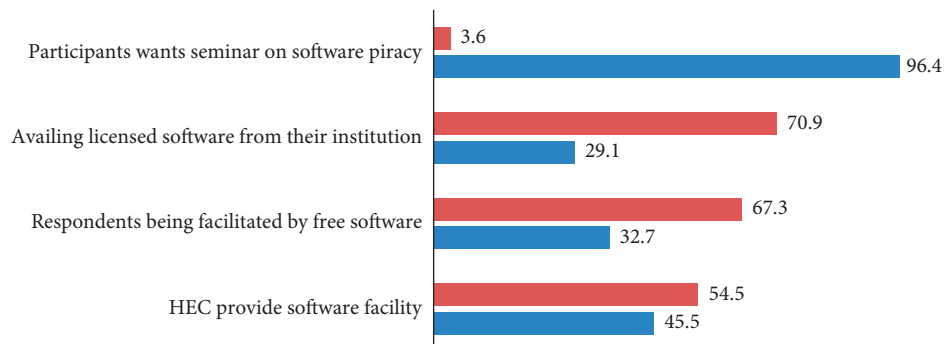


FIGURE 5: Details of the HEC policy facility.

TABLE 4: Awareness about software piracy.

S. no.	Null hypothesis	Test	Sig.	Decision
1	The categories defined by know_HEC_facility = No and yes occur with probabilities 0.5 and 0.5	One-sample binomial test	0.391	Retain the null hypothesis
2	The categories defined by institute_soft_facility = No and yes occur with probabilities 0.5 and 0.5	One-sample binomial test	0.000	Reject the null hypothesis
3	The categories defined by institute_provide_facility = No and yes occur with probabilities 0.5 and 0.5	One-sample binomial test	0.000	Reject the null hypothesis
4	The categories defined by want_seminar = No and yes occur with probabilities 0.5 and 0.5	One-sample binomial test	0.000	Reject the null hypothesis

to be implemented in the academic institutions. 90.9% of the respondent agreed to have a software policy to be implemented in academic institutions. Figure 6 shows software piracy and its causes from user perspectives.

A detailed discussion about the reasons behind software piracy is being discussed. Summary statistics are given in Table 5. We can see that difference among responses is significant for all variables (null hypothesis rejected) except the availability of software activation codes on the Internet. The reasons that retained after statistical analysis are high software cost, piracy being a way to save a significant amount of money, unavailability of online payment facilities, and poor implementation of HEC software policies in academia.

4.5. Issues Identified in the Study. Software piracy is a big issue to be considered. From the study conducted, it is obviously shown that piracy rates in academic institutions are high enough. This survey mainly includes faculty members and students from academic institutions. Although people were aware of the knowledge that piracy has several disadvantages, they do consider it an important issue and discourage piracy and also feel functionality differences between pirated and licensed software. However, there are still some issues in the awareness of software piracy. The main issues that have been identified in the survey regarding the high rate of software piracy and people awareness about software piracy are listed below:

(i) High rate of software piracy

The main issue found in the current study is the high rate of software piracy. The study aimed to focus on academic institutions where the participants are faculty members, students, and administrative staff. Still, it is observed that software piracy is high in educational institutions which are very important issues to be considered. It has been noticed that the majority of the people use pirated software and each of the faculty members uses pirated software in several different forms.

(ii) Unavailability of the online payment facility

Another issue is the availability of an online payment facility which is necessary to be available to at least faculty members of the institution. As a matter of fact, the latest and updated software are available on the Internet and mostly need online buying

procedure of purchasing. If one does not have an online payment facility, then the only choice seems to be piracy if available on the Internet because purchasing from the market is not a feasible choice. Cracks and activation codes are available on the Internet and with some searching and time spent, these can be downloaded, which is also a serious issue behind increasing software piracy.

(iii) Lack of awareness about software piracy

People are unaware of the penalties for software piracy. They only know piracy has disadvantages and licensed software provides rich functionality which is not enough. There is also a lack of knowledge about the advantages of licensed software. Similarly, almost all participants did not attend any workshop or seminar on the issue of software piracy that may lead to higher software piracy.

(iv) Poor utilization of HEC available software facilities

People know that HEC has a policy for software in academic institutions, but they do not have enough knowledge to avail their offers and benefit from it. Academic institutions do not get benefits from HEC services and we have seen in the study that these services are not facilitated for participants.

4.6. Main Reason for Issues behind Software Piracy and Awareness. We have identified some of the reasons behind software piracy. These are discussed in detail in the following subsections.

(i) Unsuitable payment methods for software purchasing

One of the reasons behind software piracy is the unavailability of online payment methods for people. Credit card or other ways are not widely being used by student(s) and all the faculty members for online transactions which forces them to use another way of registering or getting registered software.

(ii) Basic knowledge about software piracy

The people have only basic knowledge about software piracy which is not enough. The people do not

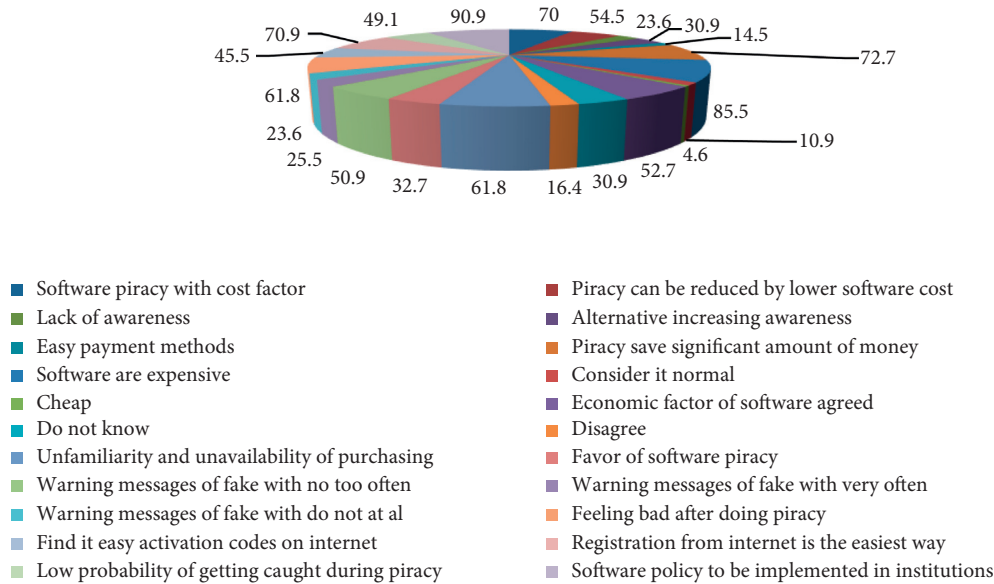


FIGURE 6: Software piracy and its causes from user perspectives.

TABLE 5: Awareness about software piracy.

S. no.	Null hypothesis	Test	Sig.	Decision
1	The categories of piracy_because occur with equal probabilities	One-sample chi-square test	0.000	Reject the null hypothesis
2	The categories of reduce_ratio_piracy_ occur with equal probabilities	One-sample chi-square test	0.000	Reject the null hypothesis
3	The categories defined by piracy_huge_money = Agree and disagree occur with probabilities 0.5 and 0.5	One-sample binomial test	0.000	Reject the null hypothesis
4	The categories of expensive_cheap occur with equal probabilities	One-sample chi-square test	0.000	Reject the null hypothesis
5	The categories of econo_incentives occur with equal probabilities	One-sample chi-square test	0.000	Reject the null hypothesis
6	The categories defined by online_buy_facility = No and yes occur with probabilities 0.5 and 0.5	One-sample binomial test	0.000	Reject the null hypothesis
7	The categories defined by favor_violation = Yes and No occur with probabilities 0.5 and 0.5	One-sample binomial test	0.000	Reject the null hypothesis
8	The categories of warning occur with equal probabilities	One-sample chi-square test	0.000	Reject the null hypothesis
9	The categories of notification_feel occur with equal probabilities	One-sample chi-square test	0.000	Reject the null hypothesis
10	The categories defined by find_active_code = Yes and No occur with probabilities 0.5 and 0.5	One-sample binomial test	0.391	Retain the null hypothesis
11	The categories of easy_soft_reg occur with equal probabilities	One-sample chi-square test	0.000	Reject the null hypothesis
12	The categories of piracy_caught_prob occur with equal probabilities	One-sample chi-square test	0.000	Reject the null hypothesis
13	The categories defined by implement_soft_policy = Disagree and agree occur with probabilities 0.5 and 0.5	One-sample binomial test	0.000	Reject the null hypothesis

know the penalties for software piracy and therefore, it seems like a normal act, although it is a crime too.

(iii) Conduction of seminar/workshops

One of the reasons for the lack of awareness includes less or lack of awareness program about software piracy. Not enough seminars or workshops have

been conducted to spread awareness about software piracy. As a result, they do not know the penalties that can be given to the person doing piracy.

(iv) Poor implementation of HEC software policies by academic institutions

The people use pirated software inside academic institutions, although the HEC provides facilities

TABLE 6: Questionnaire.

Question no.	Question description	Option (A)	Option (B)	Option (C)
<i>Demographic information</i>				
1	Age	10–25	25–35	>35
2	Gender	Male	Female	
3	Profession	Student	Faculty	Other
<i>Existing software piracy</i>				
4	Which type of software do you use?	Pirated	Licensed	
5	How many of your friends use licensed software?	All of them	Few of them	None
6	How you activate the software?	Online payment	Buying CD/DVD	Cracking it
7	Have you any facility for online payment for software?	Yes	No	
8	What are your selection criteria for choosing software?	Lower cost	Rich number of features	
9	How much pirated software do you use?	1–3	4–6	>6
<i>Awareness of software piracy</i>				
10	How much software piracy is an important issue?	Unimportant	Somehow	Very important
11	Do you think using pirated software has any disadvantages?	Yes	No	
12	Do you know about penalties for software piracy?	Yes	No	
13	Is piracy against ethics?	Yes	No	Do not know
14	Do you feel any functionality difference between pirated software and the original one?	No difference	Much difference	Do not know
15	Have you ever attended any seminar/workshop on software piracy?	Yes	No	
16	While using pirated software, the probability you will be caught is	Low	Medium	High
17	Majority of people use pirated software?	Agree	Disagree	
18	Do you feel ashamed/guilty while using pirated software?	Yes	No	Somehow
19	Sharing pirated software with others is a good act?	Yes	No	
20	Intellectual property law is beneficial for the customer	Agree	Disagree	
21	Do you know all the side effects of software piracy?	Yes	No	Somehow
<i>HEC software policy</i>				
22	Do you know about HEC software providing facilities?	Yes	No	
23	Is your academic institution providing any free software?	Yes	No	
24	Is any licensed software purchased by your institution for you?	Yes	No	
25	Are you in favor of organizing seminars/workshops on software piracy?	Yes	No	
<i>Piracy (user perspective)</i>				
26	People use pirated software because of?	Lack of awareness	High software cost	Other (please mention)
27	The ratio of software piracy can be reduced by	Increasing awareness	Decreasing software license prices	Easy payment
28	Software piracy saves a significant amount of our money?	Agree	Disagree	
29	Prices of the paid software are	Expensive	Normal	Cheap
30	The economic factor is an incentive for me to purchase pirated software	Agree	Disagree	
31	Do you have any facility for buying online software and products?	Yes	No	
32	Are you in favor of giving a violation of software piracy?	Yes	No	
33	How often you receive warnings about fake software registrations?	Very often	Few time	Never received
34	What is your feeling when you got a notification about pirated software	Normal		
35	Is it easy for you to find activation codes/cracks for software on the Internet?	Yes	No	
36	What is the easiest way to register your software?	From the Internet	From friends	Any other source
37	While using pirated software, the probability you will be caught is	Low	Medium	High
38	Employees & students need the implementation of S/W policy in an academic institution	Agree	Disagree	

of free access for some important software products.

(v) High software cost

The main and key reason is software cost. People have to get paid a significant amount of money while getting pirated software to save much of it. As a result, along with some knowledge and awareness, the people do piracy.

4.7. Proposed Suggestions to Stop/Reduce Software Piracy.

Based on the experimental study of the proposed research and results obtained, we have proposed some of the solutions/suggestions that can help in reducing software piracy. Details of each one are discussed below.

(i) Introducing suitable methods of payment for software purchasing

As most of the people do not have payment facility or do not use online payment/transaction as a primary method for purchasing. So, new methods of payment need to be adopted to reduce software piracy. It is necessary to use other methods that are available in the current market like Easypaisa, Mobi cash, and so on in the context of Pakistan as an example.

(ii) Availability of high-speed Internet in academic institutions

Internet speed is also a limiting factor. Low speed of Internet also creates problems for downloading big size of software which compels people to take from other sources instead of wasting much of their time on downloading.

(iii) Conducting and arranging seminars on software piracy in academia

Different academic programs like seminars, workshops, and training for promoted awareness about software piracy need to be initiated. In this regard, workshops or seminars to be conducted in educational institutions like universities on the highest priority. These programs should be aimed to be more focused on highlighting the bright aspects of the licensed software product.

(iv) Awareness of need-based products

On one side, people do piracy because they think the price of licensed software is much high. On the other hand, they prefer software for a high number of features. As a matter of fact, it is not necessary that software with rich features will be the best for each and every user. Each and every user has different requirements and popular software products available on the Internet have different user plans too. So, if the user is aware of his/her work needs, then he/she will pay only for these features and not for all possible features. For example, windows have

different categories like home-edition, proedition, and ultimate-edition. Prices do vary for these products based on user-specific need. Another example is Microsoft Office, which has different prices product for different user's need like students and professionals.

It is important to highlight suitable product features for specific needs. If it is done, users will download and pay for customized products with lower prices and according to their needs.

(v) HEC visits for ensuring the implementation of their policies in the academia

The HEC need visits to academic institutions to ensure how much awareness about piracy of software people have. Based on the visits to the university, the needs of software can be identified and HEC can add more software products in their plan for the future or can exclude obsolete software products that could have less importance.

(vi) Decreasing software costs and licenses prices

As one of the suggestions to attract people to licensed software mentioned above is to pay for need-based customized software product. Another good step could be to decrease software licensing prices for the user. Because despite of the awareness, people still do piracy. They claim software prices are too high.

(vii) Implementation of software policy in academia

It is important for academic institutions to get benefit from the HEC software facility. From the survey, participants are not facilitated by institutions for software products and they buy or do piracy of it by themselves.

5. Conclusions

Software piracy is an ever-increasing problem of the modern-day software industry. Owing to the evolution in software development and the Internet, software piracy has become a main concern for many software companies. Software companies are confronted with extremely high losses due to the piracy of software. Pirates gain a lot of money by doing business with pirated software, and they do not think what they are doing is a crime. General end-users and the community of the software are not well aware of this serious crime. Even most of the time, end-users and consumers think that it is none of their concern and not an important issue for them to worry about. If an organization is using pirated software, there is a risk of failure of the software, and it might put the organization into a big loss of risk. Open-source software is available, but some of this software needs a proper license from the concerned owner agencies and the user needs to pay for it. Most people cannot afford these license charges which become a burden on them. So they do piracy of the software. On the other hand, people use crack software (registered by the user through unfair way) for their needs as they do not have enough

money to pay for licensing the software, although they are aware of the real problems that pirated software have which include upgrades are not available, no assurance of quality and reliability, no technical support, no manuals or documentation, exposure of network to security breaches, and many others.

The pirated software does not receive any technical support from the organization which is developed. Due to these reasons, software piracy has turned out to be a major concern—more emergent due to the extravagant development of the software industry and the availability of software(s) on the Internet. This paper elaborates on the awareness of piracy, policy of the licensed software, and user perspective regarding the original licensed and pirated software. A questionnaire of about 38 questions was given to the students, faculty members, and administrative staff of different intuitions, and after the collection of data, analysis was performed. These questions were designed and finalized as per the discussions of the members of the project approved by the higher education commission. The results of the analysis are shown in Figures 2–6 and Table 6.

The current study identified some of the reasons for software piracy. These reasons are “unsuitable payment methods for software purchasing,” “basic knowledge about software piracy,” “conduction of seminar/workshops,” “poor implementation of HEC software policies by academic institutions,” “high software cost.”

Based on the above reasons, some suggestions are proposed by which the level of piracy can be reduced. These suggestions include “introducing suitable methods of payment for software purchasing,” “availability of high-speed Internet in academic institutions,” “conducting and arranging seminars on software piracy in academia,” “awareness of need-based products,” “HEC visits for ensuring the implementation of their policies in the academia,” “decreasing software costs and licenses prices,” and “implementation of software policy in academia.” By adopting the proposed suggestions, the level of piracy can be reduced.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare no conflicts of interest.

References

- [1] T. T. Moores and J. Dhaliwal, “A reversed context analysis of software piracy issues in Singapore,” *Information & Management*, vol. 41, no. 8, pp. 1037–1042, 2004.
- [2] L. L. Gan and H. C. Koh, “An empirical study of software piracy among tertiary institutions in Singapore,” *Information & Management*, vol. 43, no. 5, pp. 640–649, 2006.
- [3] A. Mishra, I. Akman, and A. Yazici, “Software piracy among IT professionals in organizations,” *International Journal of Information Management*, vol. 26, no. 5, pp. 401–413, 2006.
- [4] D. Curtis, “Software piracy and copyright protection,” in *Proceedings of Wescon/94: Idea/Microelectronics*, pp. 199–203, New York, NY, USA, September 1994.
- [5] R. C. Rife, “Software piracy,” in *Proceedings of Northcon/94 Conference Record*, pp. 364–366, Seattle, WA, USA, October 1994.
- [6] S. Nazir, S. Shahzad, and L. S. Riza, “Birthmark-based software classification using rough sets,” *Arabian Journal for Science and Engineering*, vol. 42, pp. 1–13, 2016.
- [7] S. Nazir, S. Shahzad, I. Zada, and H. Khan, “Evaluation of software birthmarks using fuzzy analytic hierarchy process,” in *Proceedings of the Fourth International Multi-Topic Conference*, pp. 171–175, Jamshoro, Pakistan, February 2015.
- [8] S. Nazir, S. Shahzad, Q. U. A. Nizamani, R. Amin, M. A. Shah, and A. Keerio, “Identifying software features as birthmark,” *Sindh University Research Journal (Science Series)*, vol. 47, pp. 535–540, 2015.
- [9] S. Nazir, S. Shahzad, S. A. Khan, N. Binti Alias, and S. Anwar, “A novel rules based approach for estimating software birthmark,” *The Scientific World Journal*, vol. 2015, Article ID 579390, 8 pages, 2015.
- [10] S. Nazir, S. Shahzad, and S. B. S. Abid, “Selecting software design based on birthmark,” *Life Science Journal*, vol. 11, pp. 89–93, 2014.
- [11] C. Peukert, J. Claussen, and T. Kretschmer, “Piracy and box office movie revenues: evidence from Megaupload,” *International Journal of Industrial Organization*, vol. 52, pp. 188–215, 2017.
- [12] C. J. Robertson, K. M. Gilley, V. Crittenden, and W. F. Crittenden, “An analysis of the predictors of software piracy within Latin America,” *Journal of Business Research*, vol. 61, no. 6, pp. 651–656, 2008.
- [13] S. Mumtaz, S. Iqbal, and I. Hameed, “Development of a methodology for piracy protection of software installations,” in *Proceedings of 9th International Multitopic Conference, IEEE INMIC 2005*, pp. 1–7, Karachi, Pakistan, December 2005.
- [14] J. Mo, J. Park, N. Im, J. Park, and H. Kim, “Why internet service provider and content provider do not collaborate via monitoring of digital piracy,” *Socio-Economic Planning Sciences*, vol. 60, pp. 1–13, 2017.
- [15] P. B. Lowry, J. Zhang, and T. Wu, “Nature or nurture? A meta-analysis of the factors that maximize the prediction of digital piracy by using social cognitive theory as a framework,” *Computers in Human Behavior*, vol. 68, p. 104e120, 2017.
- [16] K. S. Kumar, G. H. Rao, S. Sahoo, and K. K. Mahapatra, “Secure split test techniques to prevent IC piracy for IoT devices,” *Integration*, vol. 58, pp. 390–400, 2017.
- [17] Y.-S. Huang, S.-H. Lin, and C.-C. Fang, “Pricing and coordination with consideration of piracy for digital goods in supply chains,” *Journal of Business Research*, vol. 77, pp. 30–40, 2017.
- [18] B.-H. Chang, S.-H. Namb, S.-H. Kwon, and S. M. Chan-Olmsted, “Toward an integrated model of software piracy determinants: a cross-national longitudinal study,” *Telematics and Informatics*, vol. 34, no. 7, pp. 1113–1124, 2017.
- [19] A. Rasch and T. Wenzel, “Piracy in a two-sided software market,” *Journal of Economic Behavior & Organization*, vol. 88, pp. 78–89, 2013.
- [20] D. Banerjee, “Effect of piracy on innovation in the presence of network externalities,” *Economic Modelling*, vol. 33, pp. 526–532, 2013.
- [21] M. Siponen, A. Vance, and R. Willison, “New insights into the problem of software piracy: the effects of neutralization,

- shame, and moral beliefs,” *Information & Management*, vol. 49, no. 7-8, pp. 334–341, 2012.
- [22] A. R. Andrés and R. K. Goel, “Does software piracy affect economic growth? Evidence across countries,” *Journal of Policy Modeling*, vol. 34, no. 2, pp. 284–295, 2012.
- [23] N. K. Kariithi, “Is the devil in the data? A literature review of piracy around the world,” *The Journal of World Intellectual Property*, vol. 14, no. 2, pp. 133–154, 2011.
- [24] F. Martí’nez-Sánchez, “Avoiding commercial piracy,” *Information Economics and Policy*, vol. 22, pp. 398–408, 2010.
- [25] S. Al-Rafee and K. Rouibah, “The fight against digital piracy: an experiment,” *Telematics and Informatics*, vol. 27, no. 3, pp. 283–292, 2010.
- [26] A. Nill and C. J. Shultz, “Global software piracy: trends and strategic considerations,” *Business Horizons*, vol. 52, no. 3, pp. 289–298, 2009.
- [27] M. Peitz and P. Waelbroeck, “Piracy of digital products: a critical review of the theoretical literature,” *Information Economics and Policy*, vol. 18, no. 4, pp. 449–476, 2006.
- [28] S. N. Hamade, “The legal and political aspects of software piracy in the Arab world,” in *Proceedings of Third International Conference on Information Technology: New Generations*, pp. 137–142, Las Vegas, NV, USA, April 2006.
- [29] D. S. Banerjee, “Lobbying and commercial software piracy,” *European Journal of Political Economy*, vol. 22, no. 1, pp. 139–155, 2006.
- [30] S. H. Bae and J. P. Choi, “A model of piracy,” *Information Economics and Policy*, vol. 18, no. 3, pp. 303–320, 2006.
- [31] W. M. J. Fung and A. Lakhani, “Combatting peer-to-peer file sharing of copyrighted material via anti-piracy laws: issues, trends, and solutions,” *Computer Law & Security Review*, vol. 29, no. 4, pp. 382–402, 2013.
- [32] I. P. L. Png, “On the reliability of software piracy statistics,” *Electronic Commerce Research and Applications*, vol. 9, no. 5, pp. 365–373, 2010.