INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

RESEARCH ARTICLE

Classification and Recognition of Bilingual Text Using Graph Edit Distance Based Degree of Similarity

M J Roopa[1*], **K Mahantesh**[2]

1 Department of Computer Science and Engineering, SJB Institute of Technology, Bangalore-560060, Karnataka, India
2 Department of Electronics and Communication Engineering, SJB Institute of Technology, Bangalore-560060, Karnataka, India

OPEN ACCESS

Received: 23-12-2021
Accepted: 22-02-2022
Published: 14-07-2022

Citation: Roopa MJ, Mahantesh K (2022) Classification and Recognition of Bilingual Text Using Graph Edit Distance Based Degree of Similarity. Indian Journal of Science and Technology 15(27): 1336-1343. https://doi.org/10.17485/IJST/v15i27.2405

* **Corresponding author**.

mjroopashobith@gmail.com

Funding: None

Competing Interests: None

Published By Indian Society for Education and Environment (iSee)

ISSN
Print: 0974-6846
Electronic: 0974-5645

Abstract

Objectives: Graph Edit distance-based classification and recognition method is introduced in this study for bilingual characters. Specifically, this method aims to classify characters first and then recognize them in the 2nd level. **Methods:** This study combines both exact graph matching and inexact graph matching techniques to achieve better Recognition. The exact graph matching technique classifies characters by considering the number of vertices and edges as features to classify. Inexact graph matching uses an algorithmic model to measure the degree of similarity in the length of edit operations. **Findings:** The proposed model can compute, if not optimal, sequences all the time, at least near-optimal edit sequence. The proposed model is based on dynamic programming. Edit distance is estimated by comparing preorder sequences of the respective binary trees. Finally, this model uses the loss function to recognize text accurately. To bring out the practicality of the proposed model, we present a case study on the classification of English alphabets and Kannada conjunct consonants. **Novelty:** (i) A hybrid approach for converting a large class classification problem into a small classification problem that takes advantage of both exact and inexact graph matching techniques. (ii) An algorithmic model is designed to represent each character as a unique string. (iii) Uses dynamic programming in the right places to get the optimal tree, and (iv) to convert the graph matching problem to a string matching problem reduces the time complexity of the problem.

Keywords: Bilingual text; Character Recognition; Degree of Similarity; Graph Edit Distance; Graph Matching

1 Introduction

Graphs are more potential data structures useful for object representation in structural pattern recognition. Hence, they widely used in various domains such as computer vision and pattern recognition. Pattern Recognition and Image Processing techniques are suitable for character analysis and character recognition problems. In pattern

classification, the similarity between patterns is the key issue [1,2]. When pattern represented using graphs, the pattern matching problem turns out to be determining the similarity of graphs, which is generally known as graph matching [3–5]. A basic concept for finding the similarity between graphs is the edit distance, based on a set of edit operations [6]. The matching process is the alternative name for searching a graph or subgraph isomorphism when using graphs for pattern representation. However, real-world objects are often affected by noise such that the graph representations of identical objects may not match exactly. Thus, the matching process must include the error correction mechanism in it. Therefore, using graph edit distance errors and distortions can cope up. Refer to Fig 1 for proposed algorithmic model framework.

In the literature, various approaches have been proposed for graph matching problems. Most graph matching algorithms are based on the maximum common subgraph problem, and it is a non-deterministic polynomial-time complexity problem [7,8]. In the decision tree approach [9], model graphs are represented by decision trees. However, the trade-off is the exponential size of the tree. Neural network [10] strategies are being investigated to solve graph matching in a reasonable amount of time, but finding an appropriate neural network is a major challenge. The unifying graph matching method [11] gives a unified solution for a broad range of major graph matching problems. However, this method is transformable into the maximum clique problem. Feature calculation at the intensity level is computationally expensive; an alternative method is to use Haar features, which are very effective at detecting edges and lines in images [12]. CNN incorporates this feature in order to recognize text [13]. However, it is a slow learner when compared to other features.

From the above, it is very clear that the methods that appear to be simple are not efficient, and on the other hand, the methods that appear to be efficient are not robust. As a result, we attempted to design an efficient algorithmic model that measures the degree of similarity in the length of edit operations. This focused to combines exact graph matching and inexact graph matching techniques. The exact graph matching technique classifies characters by considering the number of vertices and edges as features to classify. Inexact graph matching measures the degree of similarity in the length of edit operations using an algorithmic model.

Furthermore, the proposed model can always compute, if not optimal, at least near-optimal edit sequences. The edit distance between two graphs is estimated by constructing maximal spanning trees of graphs. Dynamic programming is used to create the proposed model, and edit distance is calculated by comparing binary tree preorder sequences. Furthermore, this work extends to classify English, Digits, and Kannada characters. English alphabets (26 capital letters and 26 small letters), digits (10 characters), and Conjunct – Consonants (35 characters), a subset of the Kannada character set chosen for the case study. The concept of agglomerative single-link hierarchical clustering is used to classify bilingual characters. The proposed algorithmic model is sound enough at the first and second level of classifications to cluster all mutually isomorphic graphs (characters – based on the number of vertices and edges) as members of the same class and non-isomorphic graphs, as members of other classes. In the third level of classification, the maximal spanning tree and its corresponding tree traversal string template are considered to be the key to classifying/recognizing further. Extensive experimentation is conducted, and it is observed that the obtained results are encouraging.

Key contributions of this paper are listed as follows,

- A hybrid approach for converting a large class classification problem into a small class classification problem that takes advantage of both exact and inexact graph matching techniques.
- An algorithmic model is designed to represent each character as a unique string.
- Uses dynamic programming in the right places to get the optimal tree.
- Converting the graph matching problem to a string matching problem reduces the time complexity of the problem.

2 Methodology

2.1 Proposed Work

In this model, graphs are initially represented by an adjacency matrix, and then corresponding maximal spanning trees [14] is constructed. Using dynamic programming highest depth general tree, which is unique, is obtained. Forest traversal techniques [15] are applied to get equivalent binary trees, and then binary trees are converted into full binary trees by introducing dummy nodes. Subsequently, tree traversal techniques are employed to obtain the preorder and sequences of binary trees. Finally, to obtain the edit distance between binary trees corresponding preorder sequences are compared. Thus the problem of graph matching is reduced to string matching. The following subsections present detailed steps of the proposed algorithmic model.

2.2 Conversion of a graph into corresponding maximal spanning tree

In this model, graphs are represented in adjacency matrices, and then graphs are transformed into maximal spanning trees [16].

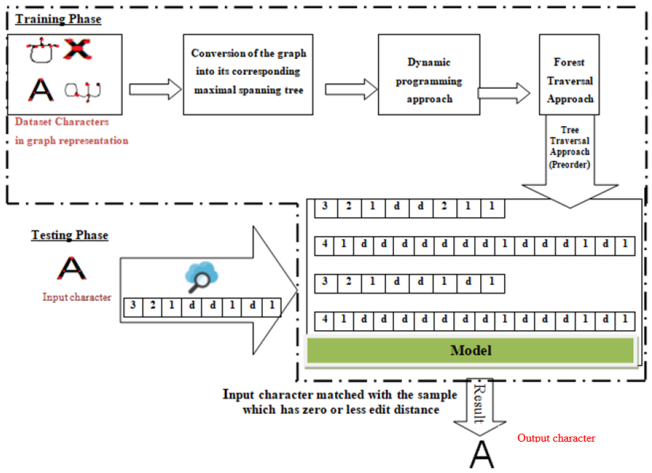

Fig 1. Framework for Proposed algorithmic model

2.3 Conversion of maximal spanning trees into general trees: Dynamic programming approach

This section introduces an algorithmic model based on the dynamic programming concept. This algorithmic model converts maximal spanning trees into general trees. The root of the general tree is the vertex having the highest degree in the graph. If there are many vertices of the same highest degree, the dynamic programming concept obtains the node whose subtree is of the highest depth. This vertex is selected to be the root of the general tree. The vertex with the next highest degree will be the leftmost child of the general tree. If there are many vertices with the same highest degree, we recommend considering the vertex as the leftmost child of the general tree with the highest degree child. Repeat the process for all the nodes of the spanning tree.

This conversion of the spanning tree to the general tree may result in more general trees corresponding to the maximal spanning tree. If such is the case, the general tree with the highest depth is selected to proceed further using the dynamic programming concept. Note that we will not generate all but only the desired one through dynamic programming.

2.4 Conversion of general trees into equivalent binary trees: Forest Traversal Approach

Transformation of binary trees into full binary trees and Computation of edit distance: Tree Traversal Approach: Conversion of the general tree into its equivalent binary tree is a 1-1 correspondence relationship. Also, the preorder traversals of binary trees are a 1-1correspondence with general trees. Thus the general tree is converted into its equivalent binary tree. This algorithmic model uses the forest traversal technique to obtain equivalent binary trees. After obtaining binary trees, the root of binary trees has always only left a subtree. Therefore dummy nodes (denoted by d) are introduced in the empty place of the left subtree of the root of the binary tree, and introducing dummy nodes to the right part of the binary trees is avoided. Thus binary trees become full binary trees.

Using the tree traversal technique, we obtained a preorder traversal sequence of full binary trees to compute edit distance. During the comparison of the preorder sequences of the binary trees, the count of edit operations increases if the first preorder sequence entry is a numerical value and the second preorder sequence is ”d” (empty place) or vice versa. The number of numerical values of the first preorder sequence corresponding to ”d” values of the second preorder sequence will give the total number of deletions while transforming the first tree into the second tree. The number of ”d” values of the first preorder sequence corresponding to numerical values of the second preorder sequence will give the total number of insertions while transforming the first tree into the second tree.

Algorithm:

Input: Object – represented in the form of Graph.

Output: Edit distance.

Method:

Step1: Classify given input using number of vertices.

Step2: Classify the classes obtained from step1 into subclasses using number of edges.

Step3: Conversion of the graph into corresponding maximal spanning tree.

Step4: Conversion of maximal spanning trees into general trees: Dynamic programming approach.

Step5: Conversion of general trees into equivalent binary trees: Forest Traversal Approach- Preorder.
Step6: Compare the input's preorder traversal to the templates, and calculate the edit distance between them.

2.5 An illustration

Input graphs are in Figure 2 (a) and (b). The output of these two graphs after applying the algorithm step 3 as shown in Figure 2 (c) and (d), respectively. The maximal spanning trees of Figure 2 (c) & (d) are then converted into the general trees, as explained in previous section. It is clear that there are two general trees for maximal spanning trees shown in Figure 2 (e), and there is only one general tree for the spanning tree shown in Figure 2 (f). As a result, we use dynamic programming to select the general trees (i) and (iii) from Figure 2 (e) with the greatest depth as the maximal spanning trees. Figure 2 (j) & (k) shows the binary trees equivalent to the general trees of (e) & (f), which are later made full binary trees by introducing dummy nodes as shown in Figure 2 (l) & (m).

The preorder sequence of the full binary tree / graph (G1) shown in Figure 2 (x) & (y) compared to find edit distance as follows:

3 3 2 1 1 1 d 1 . . . . . (array 1)
3 2 2 1 d 2 1 1 . . . . . (array 2)

The 5^{th} and 7^{th} entries in array 1 have numerical and "d" values, and the corresponding entries in array 2 have "d" values and numerical values, respectively. It implies that the deletion of one node from and addition of one node into the first tree will transform it into the second tree. Thus, two edit operations are required to transform the graph G1 to G2. Therefore edit distance is 2. This model is efficient enough to obtain the near-optimal edit sequence if not the actual edit distance without considering all possible spanning trees. Unlike the approach based on adjacency relationships which can be employed only on small size graphs, the model proposed in this topic can even be used to obtain edit distance for graphs of larger size

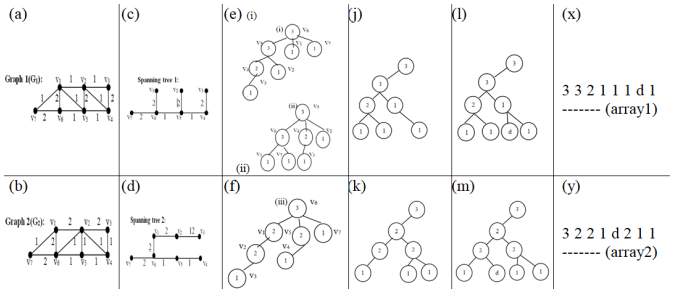

Fig 2. (a) and (b) are given graphs G1 and G2 respectively.(c) and (d) are maximal spanning trees corresponding to the graphs G1 and G2. (i)& (ii) in (e) are general trees obtained from spanning tree (c), (iii) in (f)is the general tree obtained from spanning tree (d). (j) and (k) are binary treesobtained for the binary trees Shown in (i) and (iii). (l) and (m) are fullbinary trees obtained from the general tree shown in (j) and (k), respectively

3 Results and Discussion

The suitability of the proposed edit distance approach for classifying English, Digits, and Kannada alphabets in general, conjunct consonants, a subset of Kannada alphabets in particular, was investigated.

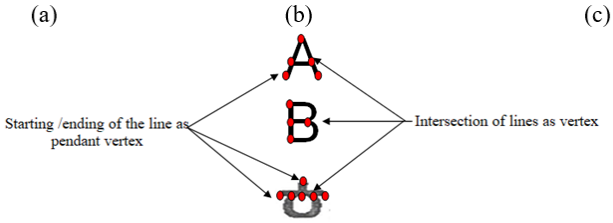

Fig 3. Characters represented using graph. (a)Example for pendant vertices (which are formed by Starting / Ending of the line), (b) Sample Characters represented in terms of vertices and edges, and(c) Example intermediate vertices (which are formed by the intersection offlines)

3.1 Experimental result on classification of English, Digits, and Conjunct Consonants

Each character represents a graph (equivalent full binary tree), and the corresponding preorder sequence is stored in the database. The characters are clustered based on the edit distance obtained for these preorder sequences. The proposed algorithm groups the closest matched characters with the shortest edit distance into a single cluster if no similar characters are found.

We have considered 52 English characters, 10 digits, and 35 Kannada conjunct consonants of a single style for experimentation. A smoothing algorithm [17] is applied initially to each character to eliminate noise from the input images (characters). Refer Figure 4 for samples.

Language	Categories & count	Characters and their Indexes (few samples)
English	Capital Letters (26 characters)	A B C D E F G H S T U V W X Y Z
	Small Letters (26 characters)	a b c d e f g h s t u v w x y z
	Digits (10 characters)	0 1 2 3 4 5 6 7 8 9
Kannada	Conjunct Consonants (35 characters)	

Fig 4. Original Characters samples taken from English and Kannada Language

Next, thinning algorithm [18] is applied to each character, which is helpful to detect the vertices of characters. The line's starting or ending is denoted as pendant vertices, and the points of intersection are taken as vertices (Figure 3). Thus, each character represents in terms of graphs. The weights of each edge are determined based on the change in direction. If the line between any two vertices is straight, then the weight is taken as one. If the line is curve linear, the weight is increased by one based on the change in direction (8 neighboring adjacency is considered). The model proposed in the previous section is applied to all 97 characters, and the computed preorder sequences of full binary trees are stored in the database. Finally, the edit distance is calculated using preorder sequences corresponding to each pair of characters. Hence, we cluster the characters into different groups based on the obtained edit distances. A proximity matrix is constructed, which consists of the computed edit distance between each pair of characters, and it is observed that this matrix is symmetric.

Observation is that the proposed model is sound enough to cluster all mutually isomorphic graphs together or members of the same class and non-isomorphic graphs as deferent classes. Figure 4 presents a few of the original characters from English and Kannada Language, and Figure 5 presents all 35 characters and their grouping along with maximal spanning trees and identical preorder sequences. Similarly, 62 (English characters and digits) classes are grouped into 7 groups based on the number of vertices, then using the number of edges each class subdivided into at most 3 subclasses. Thus 62 classes classification problem is simplified to 7 classes problem. Table 1 shows all 62 samples classes and subclasses with character index. Full binary tree traversals of the English characters samples are shown in Figure 6.

Classes	Characters index	Graph representation of the characters	Sub-classes	Subclass Graphs	Maximal spanning trees	Preorder traversal of the full binary tree
C1	(12), (18)		-	-		1
C2	(16)(19)(3)(27) (32)(26)(33)		C21			1 1
			C22			1 1
			C23			
C3	(2) (13) (20)(30) (25)(23)(35) (17)		C32			1 1
			C33			1 1 2 1 d 1
C4	(21)(28) (14)(34)		C43			3 1 d d̲ d̲ 1 d 1
			C44			3 1 d d̲ d̲ 1 d 1 2 2 1 1
C5	(31) (8)(15)(5) (7)(22)(24)(9)		C54			3 2 1 d d̲ 1 d 1 4 1 d d̲ d̲ d̲ d̲ d̲ 1 d d̲ d̲ 1 d 1
			C55			3 2 1 d d̲ 1 d 1
						4 1 d d̲ d̲ d̲ d̲ d̲ d̲ 1 d d̲ d̲ 1 d 1
C6	(1) (29) (10)(6)		C66			3 2 1 d d̲ 2 1 1
			C65			4 2 1 d d̲ d̲ d̲ d̲ d̲ 1 d d̲ d̲ 1 d 1
C7	(4)		-	-		3 3 1 d d̲ d̲ 1 d d̲ 1 d d̲ d̲ 1 d 1
C8	(11)		-	-		2 2 2 1 d 1 1 d d̲ 1 d d̲ d̲ 1 d d̲

Fig 5. Kannada characters -35 grouped into 8 classes and their equivalent full binary tree traversal/string representation of all characters

Table 1. English and digits - 62 characters are grouped into 7 classes and their subclasses with character index

Classes	Classified Characters index (Refer Table1)	Sub- classes	Classified Characters index (Figure 4) for subclass
C0	(15), (27), (51)	-	-
C1	(35)	-	-
C2	(3), (4), (9), (10), (19), (21) (33), (34), (36), (39), (48), (55)	C21	(3), (9), (10), (19), (21), (39), (48), (55)
		C22	(4), (33), (36)
C3	(12), (16), (17), (22), (28), (29), (30), (34), (37), (41), (43), (45), (46), (58)	C31	(45), (46)
		C32	(12), (22), (28), (29), (30), (34), (58)
		C33	(16), (17), (37), (41), (43)
C4	(2), (7), (14), (20), (25), (26), (32), (38), (40), (44), (50), (52), (53), (54), (57), (61), (62)	C43	(7), (14), (20), (25), (26), (32), (44), (50), (54), (57), (61), (62)
		C44	(38), (40), (52), (53)
		C45	(2)
C5	(1), (6), (13), (18), (23), (24), (31), (42), (56), (59), (60)	C54	(6), (13), (23), (24), (42), (56), (59), (60)
		C55	(1), (18), (31)
C6	(5), (8), (11), (47), (49)	-	-

Classes	Characters index	Graph form of the characters	Subclasses	Graph form of the subclass characters	Maximal spanning trees	Preorder traversal of the full binary tree
C2	(3)		C21			1 1
C3	(43)		C33			2 1 d 1
C5	(60)		C54			4 1 d d d d d d d d 1 d d d 1 d 1

Fig 6. Some sample examples of English characters with equivalent tree traversal of full binary tree/string representation

The proximity matrix in Table 2 contains a few sample characters for showing the edit distance between each pair of characters. Because the proximity matrix is symmetric, the diagonal values are always zero, implying that the edit distance between the same characters is zero. Furthermore, the lower diagonal matrix values are the same as the upper diagonal matrix values. The edit distance between characters (3) and (43) is two, which means that we need two edit operations to convert character 'c' to 'g,' namely, one vertex addition and one edge addition. Other edit operations between the two characters are similar.

Table 2. Proximity matrix to represent edit distance between sample characters

English Character index	(3)	(43)	(60)
(3)	0	2	6
(43)	-	0	4
(60)	-	-	0

3.2 Loss Function

The loss function defined for our model is good enough to recognize characters by considering the weight of the deleted edges during the conversion of the graph into the spanning-tree process as one of the features. This function helps in recognizing characters when string representations of two different characters are the same. String representation of the graph index (16) and (19) are the same, and hence graph edit distance between them is zero. In this case, only classification is successful, but recognition fails in some cases. Therefore, the loss function plays a major role in successfully recognizing character by considering the deleted edge's weight. Refer Figure 7. The Loss function for our model is defined as follows,

Loss function $(L_f) = L_{we'} + L_{we}$ (1)

Where $L_{we'}$ is the sum of the all deleted edges weight during graph to spanning tree conversion and L_{we} is the sum of the weight of all edges of the spanning tree. The recognition rate for bilingual characters is as shown in Figure 8.

Table 3 demonstrates how the Neural network [12] with Haar features achieves 96.61 % accuracy. Another approach by the same author is the Radial basis function with haar features, which yields 95.66 %, and the back propagation with haar features,

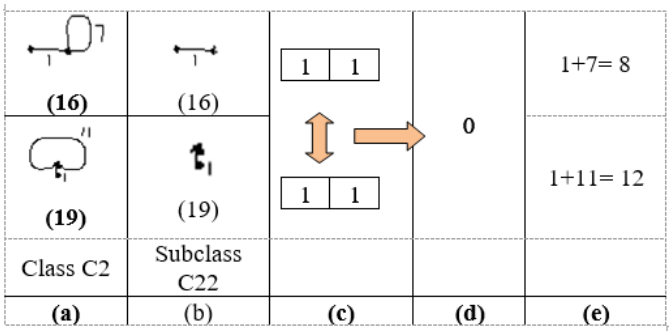

Fig 7. Importanceof Loss function (Le) in Recognition of text. (a) Kannada dataset sample graphsin class C2. (b) Maximal Spanning tree with weight obtained for the graphsconsidered in C2 class (c) Full binary tree preorder traversal to obtain astring representation of the respective graph. (d) The computed Graph EditDistance between both sequences of (c). (e) Weight of the graph by consideringthe weight of the maximal spanning tree and Lwe weight.

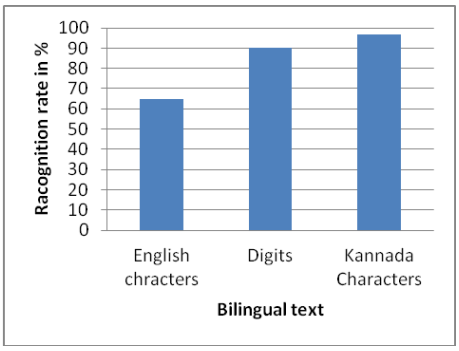

Fig 8. Recognition rate of English, Digits, and Kannada Characters

which yields 95.10 %. This demonstrates that combining haar features and neural networks works well to achieve a higher recognition rate. Discrete cosine transform (DTC) features are used in all three methods, but neural network and DTC work better than other combinations. However, our proposed algorithmic model outperforms other methods based on basic graph features and loss functions defined in the preceding subsection.

Table 3. Recognition Accuracy rate for Kannada Conjunct Consonants

Method	Features	Recognition Rate (%)
Neural Network [12]	Haar	96.61
Radial Basis Function [12]	Haar	95.66
Back Propagation Network [12]	Haar	95.10
Neural Network [12]	Discrete Cosine Transform	96.79
Radial Basis Function [12]	Discrete Cosine Transform	95.48
Back Propagation Network [12]	Discrete Cosine Transform	93.78
Proposed Algorithmic Method	Graph features	97.01

4 Conclusions

In this study, an algorithmic model is designed to classify and recognize bilingual text. It is a hybrid technique. Initially, it applies the exact graph matching technique to simply classification, and then later inexact graph matching finally loss function is used to recognize text accurately. Proposed hybrid approach converts large class classification problem into a small class classification problem that takes advantage of both exact and inexact graph matching techniques. The inexact graph matching method determines the edit distance between graphs represented in terms of binary trees. The proposed system uses forest

traversal and tree traversal techniques to compute edit distance in string matching of traversal sequences. The application of the proposed model to classify bilingual text that is English and Kannada conjunct consonants are explored, and the proposed model is smart enough to find the correct clusters as well as accurate Recognition. The benefit of this algorithmic model is that it allows the user to search for a desired character/graph in a database that contains string representations of the character/graph. It uses dynamic programming in the right places to get the optimal tree. This represents each character as a unique string. As a result, this model becomes general and adaptable framework for searching similar graphs (characters) and it can be extended to recognize any language text. Finally this algorithmic model represents each character as a unique string. Converting the graph matching problem to a string matching problem reduces the time complexity of the problem.

References

1) Adel D, Ibrahim C, Saïd Y. Efficient approximate approach for graph edit distance problem. *Pattern Recognition Letters (IF3756)*. 2021;151:310–316. Available from: https://doi.org/10.1007/s10618-020-00733-5.
2) Ma G, Ahmed NK, Willke TL, Yu PS. Deep graph similarity learning: a survey. *Data Mining and Knowledge Discovery*. 2021;35(3):688–725. Available from: https://doi.org/10.1109/ACCESS.2020.2993191.
3) Osman AH, Barukub OM. Graph-Based Text Representation and Matching: A Review of the State of the Art and Future Challenges. *IEEE Access*. 2020;8:87562–87583. Available from: https://doi.org/10.1109/ACCESS.2020.2993191.
4) Ok S. A graph similarity for deep learning. *Advances in Neural Information Processing Systems, 2020-December(NeurIPS*. 2020. Available from: https://papers.nips.cc/paper/2020/file/0004d0b59e19461ff126e3a08a814c33-Paper.pdf.
5) Raveaux R. On the unification of the graph edit distance and graph matching problems. *Pattern Recognition Letters*. 2021;145:240–246. Available from: https://doi.org/10.1016/j.patrec.2021.02.014.
6) Serratosa F. Redefining the Graph Edit Distance. *SN Computer Science*. 2021;2(6):1–8. Available from: https://doi.org/10.1007/s42979-021-00792-5.
7) Rica E, Álvarez S, Serratosa F. Ligand-Based Virtual Screening Based on the Graph Edit Distance. *International Journal of Molecular Sciences*. 2021;22(23):12751–12751. Available from: https://doi.org/10.3390/ijms222312751.
8) Marcelli A, Quer S, Squillero G. The Maximum Common Subgraph problem: A portfolio approach. 2019. Available from: https://arxiv.org/pdf/1908.06418.pdf.
9) Harsh P, Purvi P. Study and Analysis of Decision Tree Based Classification Algorithms. *International Journal of Computer Sciences and Engineering*. 2018;6(10):74–78. Available from: https://doi.org/10.26438/ijcse/v6i10.7478.
10) Chawhan RB, Rao MR. Graph Based Printed Kannada Character Recognition System By Using Backpropagation Neural Network. 2020;11:418–424.
11) Jiang Z, Wang T, Yan J. Unifying Offline and Online Multi-Graph Matching via Finding Shortest Paths on Supergraph. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2021;43(10):3648–3663. Available from: https://doi.org/10.1109/TPAMI.2020.2989928.
12) Kumar V, Ramakrishnan B, G A. Machine recognition of printed Kannada text. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics*. 2002;p. 37–48. Available from: https://doi.org/10.1007/3-540-45869-7_4.
13) Hazra A, Choudhary P, Inunganbi S, Adhikari M. Bangla-Meitei Mayek scripts handwritten character recognition using Convolutional Neural Network. *Applied Intelligence*. 2021;51(4):2291–2311. Available from: https://doi.org/10.1007/s10489-020-01901-2.
14) Chen Z, Xie Z, Yan J, Zheng Y, Yang X. Layered Neighborhood Expansion for Incremental Multiple Graph Matching. *Computer Vision – ECCV 2020*. 2020;p. 251–267. Available from: https://doi.org/10.1007/978-3-030-58607-2_15.
15) Bhagwat G, Kumari S, Patekar V, Deaconu AM. Novel Static Multi-Layer Forest Approach and Its Applications. *Mathematics*. 2021;9(21):2650–2650. Available from: https://doi.org/10.3390/math9212650.
16) Sheikh IA, Shahid I. Minimum Spanning Tree Algorithms and and Techniques. 2018. Available from: http://www.ijciras.com/PublishedPaper/IJCIRAS1101.pdf.
17) Sann SS, Win SS, Thant ZM. An analysis of various Image Pre-Processing techniques in Butterfly Image. *International Journal of Advance Research*. 2021;6(1).
18) Dey R, Balabantaray RC, Mohanty S. Sliding window based off-line handwritten text recognition using edit distance. *Multimedia Tools and Applications*. 2021. Available from: https://doi.org/10.1007/s11042-021-10988-9.