DOI: 10.15849/IJASCA.230320.12

Int. J. Advance Soft Compu. Appl, Vol. 15, No. 1, March 2023
Print ISSN: 2710-1274, Online ISSN: 2074-8523

High-Performance Computing of Building The Dependency Trees and Calculating Tree Edit Distances For Text Similarity

Nesreen Alsharman, Inna V.pivkina, Raja M.T Masadeh , Omar Almomani , and Nabeel Bani-Hani
Computer Science Department, The World Islamic Sciences Education University, Amman, Jordan
e-mail: Nesreen.alsharman@wise.edu.jo
Computer Science Department, New Mexico State University, USA
e-mail: ipivkina@cs.nmsu.
Computer Science Department, The World Islamic Sciences Education University, Amman, Jordan
e-mail: raja.masadeh@wise.edu.jo
Department of Information Systems and Network, The World Islamic Sciences Education University, Amman, Jordan
e-mail: omar.almomani@wise.edu.jo
University of Bahrain, College of Applied Studies
e-mail: nbanihani@uob.edu.bh

Abstract

Calculating similarity in text is a fundamental task in many natural language processing studies such as text summarization and it is still work in progress. In this paper, we propose a high-performance computing approach for building dependency trees and calculating the Tree Edit Distance technique (TED) for determining the similarity to links of dependency for n-gram syntactic and grammatical that shows the soft similarity between a set of related sentences text. Building dependency trees and determining tree edit distances for a large data set takes a lot of time, but creating a dependency tree for one sentence is independent of creating a dependency tree for another sentence. For example, generating a dependency tree for sentence three does not depend on generating a dependency tree for sentence four furthermore the edit distance computation between two dependency trees of sentences is independent of the computation of the edit distance between two other dependency trees of sentences. These operations calculating TED and creating sentence dependency trees can be implemented in parallel using multiprocessing which could greatly reduce the time.

Keywords: *Tree Edit Distance, High-Performance Computing, Text Similarity.*

1 Introduction

Finding the similarity between a group of related sentences is a fundamental step or stage in several Natural Language Processing (NLP) application areas, including Information Retrieval, Question Answering (QA) response selection, Text summarization, Plagiarism

Detection, Paraphrase Identification (PI), Recognizing Textual Entailment (RTE), and many other NLP tasks [4, 2, 3, 25, 19, 13,31,32] rely on text similarity. In general, the methods of finding the similarity between a set of related sentences depend on the representation of sentences. There are two ways for sentence representation: a syntactic representation of sentences and a bag-of-words representation. The structure of the text can be expressed using syntactic representation, which can also discriminate between events and their surroundings in the text. The same entities or events can be expressed in numerous viewpoints using a broad set of relations between the events and their participants. With these features, a dependency tree is constructed for each sentence in the text and one of the problems with the bag-of-words model is solved in that it ignores the relationship between the words in a sentence. We use the following example to demonstrate the issue:

1) "Ahmad gets Ali's vote.
2) " 2) "Ali gets Ahmad's vote."

Both statements are identical in the bag-of-words representation. The statements, however, are distinct according to dependency tree relations since they have various tree representations. Also, the bag-of-words model fails to capture structural relationships and PoS classification. Consider the following two scenarios. "Ahmad and Ali count votes," and "Ahmad votes for Ali," for example. The word "votes" is a noun in the first sentence but a verb in the second. In this paper, TED depends on dependency trees to represent grammatical and syntactic relationships between words to compute similarity to solve the bag-of-words problem. For a huge set of related sentences collection. It takes a long time to calculate tree edit distances and construct dependency trees. However, constructing a dependency tree for one sentence does not rely on creating one for constructing (for example, constructing a dependency tree for statement 2 does not construct a dependency tree for sentence 1). Furthermore, determining the tree edit distance between two sentences is independent of determining the tree edit distance between two additional sentences. These tasks (constructing sentence dependency trees and calculating TED) can be completed simultaneously (completed in parallel).

2 Related Work

In algorithmic research, calculating TED [1, 15, 12, 18, 14, 5, 28] has received a lot of attention. For example [1] indicates that one of the most fundamental activities of bioinformatics, such as networks of phylogenetic RNA 2 secondary structures evaluating glycans, uses TED extensively when comparing structured tree data. In addition, Fukagawa in [12] offered a method for calculating the edit distance among unordered rooted trees that were both appealing and practical. The problem of estimating the distance across unordered trees is transformed into the maximum clique problem for locating RNN glycan similarity structures. Recently, TED-based methods have been increasingly popular in NLP assignments [26, 21, 10]. For example, the TED research by kouylekov and magnini [17] is designed to identify textual entailment through dependency trees. The authors also examine various approaches to calculating the edit distance algorithms' cost functions. Also, in NLP tasks such as research for answering question systems and text entailment [27] produced structured latent variables with

probabilistic models for tree editing. The tree edits conditional random field model has been used by the authors of this study to assess the semantic similarity between phrase pairs. The generalized TED was used by the authors to demonstrate how to embed alignments, such as structured latent variables, in a probabilistic model.

3 The Proposed Method

In Natural Language Processing (NLP) applications such as Paraphrase Identification (PI), Plagiarism Detection and other domains linked to text processing, such as Information Retrieval, the computation of text similarity is a fundamental operation. In general, there are two ways for calculating text similarity: strings comparison (bag of words comparison) and syntactic n-grams (syntactic tree comparison). Some algorithms, such as LexRank [22], treat sentences as collections of words. This depiction may miss certain linguistically related information, decreasing the quality of the summary. For summarizing, some researchers use a supervised probabilistic model over syntactic trees to address the bag-of-words representation (e.g. [16]). Others use semantic frames and syntactic dependencies to tackle this issue, which indicate significant semantic and grammatical links in the text. [23]. Figure 1 shows the proposed method.

The preprocessing steps in this research are: tokenization, segmentation, and extracting semantic and syntactic dependencies using the Stanford Dependency parser [20], then building a dependency tree for each sentence. Figure 2 is an example of Stanford's semantic and syntactic dependency relations. Figure 3 is the dependency tree for the sentence in figure 2.

After preprocessing steps, calculating multiprocessing of TED is performed for finding text similarity that handles the bag-of-words problem. But in the previous related work, Dependency parsing was formerly used to locate common information among sentences to achieve sentence fusion [6, 11] and for sentence compression, TED was used to recognize uninformative sections of sentences [29]. TED is the most popular calculating model for finding the similarity between structures in natural language processing and information retrieval. In big data, calculating the tree edit distance between two sentences and constructing grammatical trees take a long time. On the other hand, calculating the tree edit distance between two sentences is unrelated to calculating the tree edit distance between two further sentences so these tasks (constructing dependency trees for a set of sentences and calculating TED) can be done at the same time in parallel to reduce time complexity and do more NLP takes in faster. Finally, multiprocessing for calculating the summary is performed. In general, there are two methods for generating a summary: extractive and abstractive summarization [27, 24]. The language in the document is initialized by abstractive summarization, and if necessary, additional words or phrases are added to the summary. Extractive summarization constructs a summary using only a part of the text's statements. This research implemented two types of summarizations. The proposed clique abstractive method consists of the following steps: After choosing a smaller group of candidate sentences that have the most common grammatical relations,

the main step is to generate a sentence similarity graph, in which its nodes representing sentences and TED scores indicate the similarities between vertices (unigrams and bigrams). The second step is to search the similarity sentence graph for all cliques subgraphs, which are complete subgraphs of a graph with each node related to every other. In the second extractive clique approach, we pick the single sentence from each cluster with the lowest cluster-specific TED similarity average.

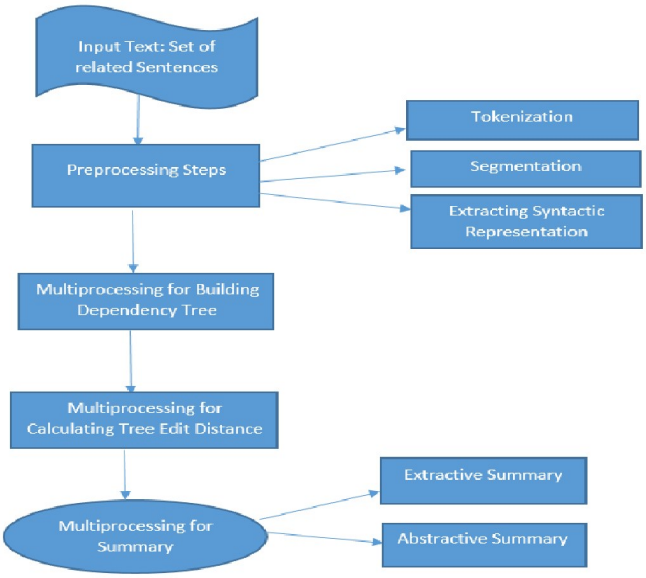


Figure 1: Proposed Method.

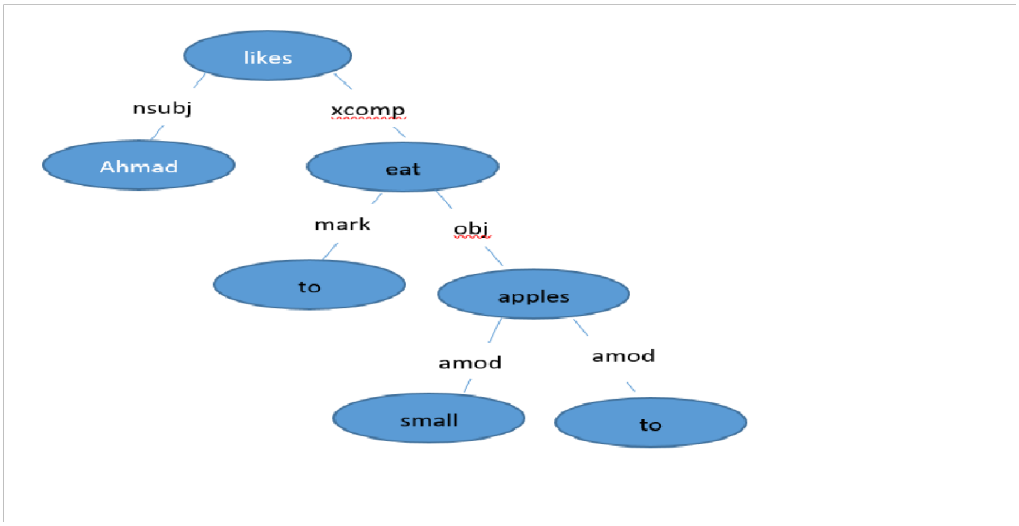


Figure 2: Stanford Semantic and Syntactic Dependency Relations.

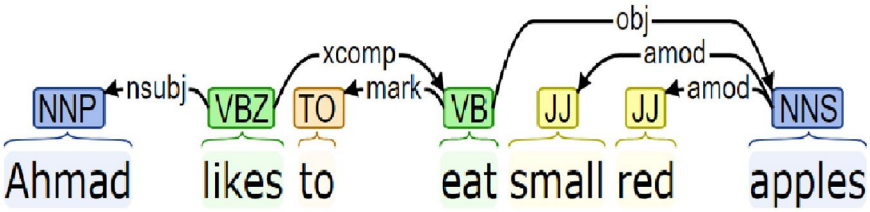


Figure 3: Stanford Semantic and Syntactic Dependency Relations

4 High-Performance Computing for Building Dependency Trees

Before constructing dependency trees, the Stanford dependency grammar representation has been retrieved for all sentences in the text. These representations comprise grammatical and syntactic relationships between words in a sentence. From these relations, the dependency tree for each sentence is built to find the similarity of sentences. A dependency tree T is a data structure generated from a set of nodes $N(T)$ and edges E(T) $\subseteq N(T) \times N(T)$. T should have the following constraints:

1) T does not have any cycles and every node has at most one incoming edge.
2) T has a single root that is a node without incoming edges
3) every node of T has just one incoming edge
4) a path from the root to each node $N(T)$ is unique.

We keep a queue of nodes (words) to process while creating the dependency tree. A queue is initially empty. To pick up the root node of the dependency tree, start with the root relation. Create a queue for the root node. Second, delete node X from the top of the queue, discover all words related to the node from relations, and add them as children (as new nodes) to the node with edges that show the types of relations. Add all newly added nodes to the queue's end. Third, repeat the second process recursively for new words (nodes) until the tree is complete.

Since building the dependency tree for a specific sentence does not depend on another dependency tree, the multiprocessing task using Python is used to achieve high performance and reduce time complexity. Figure 4 shows python functions that describe the multiprocessing task for building a dependency tree.

5 High-performance computing for calculating TED and finding text similarity

The creation of a list of candidate sentences is the stage that is most crucial in the summarizing process. The candidate sentences are valuable when they are represented as a collection of dependency trees because these representations show semantic and syntactic relationships that are valuable for summarization. The candidate sentences in this study are represented as 5 a collection of dependency trees, then TED is used to determine the degree of similarity between sentences. Zhang and Shasha in 1989 developed an algorithm for choosing the optimal matching tree solution for a pair of trees, whereas Tai in 1979 proposed a criterion for matching nodes between tree representations.

The TED algorithm is dependent on ordered labeled trees. Ordered labeled trees are those in which the relationship between siblings is important, starting from the left to the

right. The difference in tree edit costs between two trees is known as the tree edit distance. The following edit operations are used: removing a node and linking its children to its parent while maintaining order, inserting a node between an existing node and a series of this node's subsequent.

```
def StartProcessing():
    global  parsed_sentences_list
    global sent_distances_list
    All_Sentences = filereader.ReadFiles('NYT19981002.0322')
    sent_id = 0
    jobs = []
    for s in All_Sentences:
        sentence = clsSentence()
        q = multiprocessing.Queue()
        sentence.sentence_id =  sent_id
        sent_id += 1
        sentence.sentence_text = s.strip()
        pro = multiprocessing.Process(target=_ParseSentence,args=( sentence.sentence_text,q) )
        pro.start()
        sentence.relations_str = q.get()
        sentence.relations_list = q.get()
        sentence.dep_tree = q.get()

        jobs.append(pro)
        parsed_sentences_list.append(sentence)

    for p in jobs:
        p.join()
        # sentence.dep_tree.show(tree_style=ts)
    sent_distances_list = []
    most_frequenet_list = clsmost_freq_rels.Most_Frequent_Sentnces(parsed_sentences_list)
    # parsed_sentences_list = [x.sentences for x in most_frequenet_list]
    parsed_sentences_list=[]
    for s in most_frequenet_list:
        for sent in s.sentences:
            sentence = clsSentence()
            sentence.sentence_id = sent.sentence_id
            sentence.sentence_text = sent.sentence_text
            sentence.dep_tree = sent.dep_tree

            parsed_sentences_list.append(sentence)
```

Figure 4: Function that Describes the Multiprocessing for Building a Dependency Tree.

children, and renaming a node's label. The cost of each operation is determined by a cost function γ. An edit script S between two trees T1 and T2 is a sequence of edit operations turning T1 into T2. The total cost of all the operations in S is the cost of S. The most popular dynamic programming algorithm for calculating the edit distance among two ordered trees is the Zhang-Shasha method [30].

Table 1: F-ROUGE-Average Measures

ROUGE Average	Lex-Rank	Cliques Abstractive Method	cliques Extractive Method
ROUGE-1 :	0.45	0.55	0.58
ROUGE-2 :	0.24	0.38	0.40
ROUGE-3 :	0.18	0.28	0.34
ROUGE-4 :	0.15	0.19	0.30
ROUGE-L :	0.39	0.51	0.52
ROUGE-W :	0.12	0.21	0.25
ROUGE-SU :	0.20	0.25	0.29

6 Results and dissection

The proposed method was evaluated on DUC 2004 (Task 4 and Task 3) data set [9]. The data set contains news documents in English together with human-generated summaries for the documents. We selected DUC 2004 data set because it has been used in many previous NLP-related works, e.g., [10, 7].

The ROUGE Perl tool [8] was used to compare the automatically generated summary to the manually generated summary. Many additional researchers employed ROUGE to analyze their summarization algorithms. In ROUGE, we choose the option to use Porter's stemmer on the input.

We give each evaluation metric an F-score. F-ROUGE will offer an accurate comparison because the summary length is not rigidly enforced to compare automatically generated summaries to a group of reference summaries. Table 1 shows the comparison between extractive clique summary, abstractive summary, and LexRank.

TED is sometimes overlooked because of its inefficiency, despite its simplicity and relevance. The time complexity of TED is cubic and the space complexity is quadratic because comparing trees of one million nodes may require (50-100) hours of runtime and 1.2TB of memory). Distributing a computation across many CPU cores is one technique to boost the performance of an algorithm. The tree edit distance, on the other hand, is far from straightforward to parallelize. The results of smaller problems are utilized to calculate the values for larger problems in dynamic programming. The dependency between these values must be taken into account to parallelize the algorithm. In this research, the average time complexities are reduced from (50-100) hours to (2- 5) minutes.

Conclusion

The main features in this research are utilizing high-performance computing for building dependency trees to reduce the time for calculating text similarity and utilizing syntactic trees to tackle the representation of a bag of words and to express grammatical and syntactic relationships. According to Table 1, the results of cliques of the abstractive method are better LexRank and cliques abstractive methods.

References

[1] Tatsuya AKUTSU. "Tree Edit Distance Problems: Algorithms and Applications to Bioinformatics". In: *IEICE Transactions on Information and Systems* E93.D.2 (2010), pp. 208–218. doi: 10.1587/transinf.E93.D. 208.

[2] Nesreen Alsharman et al. "Machine Learning and Answer Set Program Rules towards Traffic Light Management". In: *International Journal of Advanced Trends in Computer Science and Engineering* 9.3 (2020).

[3] Wael Jumah Alzyadat et al. "A Recruitment Big Data Approach to interplay of the Target Drugs". In: *International Journal of Advances in Soft Computing and its Applications* (2022).

[4] FAISAL Y. ALZYOUD, NISREEN ALSHARMAN, and ABDALLAH ALTAHAN ALNUAIMI. In: *Journal of Theoretical and Applied Information Technology*. 2021, pp. 38–47.

[5] "An Efficient Unordered Tree Kernel and Its Application to Glycan Classification". In: *Advances in Knowledge Discovery and Data Mining* (2008).

[6] Regina Barzilay and Kathleen R. McKeown. "Sentence Fusion for Multidocument News Summarization". In: *Comput. Linguist.* (2005), pp. 297– 328.

[7] Yllias Chali and Sadid A. Hasan. "Query-focused multi-document summarization: Automatic data annotations and supervised learning approaches". In: *Natural Language Engineering* (2012), pp. 109–145.

[8] Arman Cohan and Nazli Goharian. "Revisiting Summarization Evaluation for Scientific Articles". In: (2016).

[9] *DUC. Proceedings of Document Understanding Workshop (DUC 2004)*. Boston Park Plaza Hotel and Towers, Boston, USA, May 6-7, 2004. 2004.

[10] Gunes Erkan and Dragomir R. Radev. "LexRank: Graph-based Lexical Centrality As Salience in Text Summarization". In: *Journal of Artificial Intelligence* (2004), pp. 457–479.

[11] Katja Filippova and Michael Strube. "Sentence Fusion via Dependency Graph Compression". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2008, pp. 177–185.

[12] Daiji Fukagawa et al. "clique-based method for the edit distance between unordered trees and its application to analysis of glycan structures." In: *BMC Bioinformatics* (2011).

[13] Adnan Hnaif, Emran Kanan, and Tarek Kanan. "Sentiment Analysis for Arabic Social Media News Polarity". In: *Intelligent Automation and Soft Computing* (2021).

[14] Yair Horesh, Ramit Mehr, and Ron Unger. "Designing an A* algorithm for calculating edit distance between rooted-unordered trees". In: *J Comput Biol* (2006), pp. 1165–76.

[15] Tao Jiang et al. "A general edit distance between RNA structures". In: *Journal of Computational Biology* 9 (2002), pp. 371–388.

[16] Kevin Knight and Daniel Marcu. "Statistics-Based Summarization – Step One: Sentence Compression". In: *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*. 2000, pp. 703–710.

[17] Milen Kouylekov. "Recognizing Textual Entailment with Tree Edit Distance: Application to Question Answering and Information Extraction". In: 2006.

[18] Bin Ma, Lusheng Wang, and Kaizhong Zhang. *Computing similarity between RNA structures*. 2002.

[19] Ginika Mahajan and Neha Chaudhary. "Improving Bug Localization using IR-based Textual Similarity and Vectorization Scoring Framework". In: *International Journal of Advances in Soft Computing and its Applications* (2020).

[20] Marie-catherine De Marneffe and Christopher D. Manning. “The Stanford Typed Dependencies Representation”. In: *Coling 2008: Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation*. 2008, pp. 1–8.

[21] Rada Mihalcea and Paul Tarau. “A language independent algorithm for single and multiple document summarization”. In: *In Proceedings of IJCNLP’2005*. 2005.

[22] Saziye Betul Ozates, Arzucan Ozgur, and Dragomir Radev. “Sentence Similarity based on Dependency Tree Kernels for Multi-Document Summarization”. In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation* (2016).

[23] Natalie Schluter and Anders Sogaard. “Unsupervised extractive summarization via coverage maximization with syntactic and semantic concepts”. In: *ACL* (2015).

[24] Chao Shen and Tao Li. “Multi-document Summarization via the Minimum Dominating Set”. In: *Proceedings of the 23rd International Conference on Computational Linguistics*. 2010, pp. 984–992.

[25] Ahmad T. Al-Taani. (2021) “Recent Advances in Arabic Automatic Text Summarization”. In: *International Journal of Advances in Soft Computing and its* Applications 13(3):59-71.

[26] Kapil Thadani. “Multi-Structured Model for Transforming and Algning Text”. In: *COLUMBIA UNIVERSITY*. 2015.

[27] Mengqiu Wang and Christopher Manning. “Probabilistic Tree-Edit Models with Structured Latent Variables for Textual Entailment and Question Answering”. In: *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. Beijing, China: Coling 2010 Organizing Committee, Aug. 2010, pp. 1164–1172.

[28] Yoshihiro Yamanishi, Francis Bach, and Jean-Philippe Vert. (2007) “Glycan classification with tree kernels”. In: *Bioinformatics* 23 (10), pp. 1211–1216.

[29] Mehdi Yousfi-Monod and Violaine Prince. “Sentence Compression as a Step in Summarization or an Alternative Path in Text Shortening”. In: *Coling’08: International Conference on Computational Linguistics*. 2008, pp. 139–142. url: https://hal-lirmm.ccsd.cnrs.fr/lirmm-00318727.

[30] K. Zhang and D. Shasha. “Simple Fast Algorithms for the Editing Distance Between Trees and Related Problems”. In: *SIAM J. Comput.* 18.6 (1989).

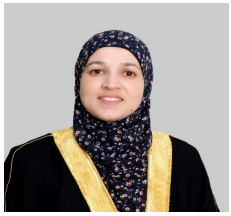
[31] Mohammad, A. H., Alwada’n, T., & Al-Momani, O. (2016). Arabic text categorization using support vector machine. Naïve Bayes and Neural Network. GSTF Journal on Computing (JoC), 5, 1-8.

[32] Ababneh, J., Almomani, O., Hadi, W., El-Omari, N. K. T., & Al-Ibrahim, A. (2014). Vector space models to classify Arabic text. International Journal of Computer Trends and Technology (IJCTT), 7(4), 219-223.

[33] Mohammad, A. H., Almomani, O., & Alwada'n, T. (2016). Arabic Text Categorization Using k-nearest neighbour, Decision Trees (C4. 5) and Rocchio Classi¯ er: A Comparative Study. International Journal of Current Engineering and Technology, 6(2).

***Nesreen Alsharman*,** works as an assistant professor at World Islamic Sciences and Education University (Department of Computer Science). Nesreen conducts research in the areas of big data classification, artificial intelligence, traffic congestion, and natural language processing (text summarization and text classification).

Inna V. Pivkina works as an Associate Professor of Computer Science, at New Mexico State University. Data mining applications, computer science education, and natural language processing are some of Inna Pivkina's areas of interest in her research.



Raja Masadeh works as an assistant professor at the World Islamic Sciences and Education University, department of Computer Science. Raja conducts research in the area of artificial intelligence, optimization algorithms, cloud computing and Internet of Things.

Omar Almomani received his Bachelor's and Master's degrees in Telecommunication Technology from the Institute of Information Technology - at the University of Sindh in 2002 and 2003 respectively. Almomani received his Ph.D. from UNIVERSITY UTARA MALAYSIA in computer network and security in 2010. Currently, he is Professor in and Information Systems and Network Department, Information Technology Faculty at the World Islamic Sciences & Education University. His research interests involve Network Performance, Network Quality of Service (QoS), IoT, Network Modeling and Simulation, Network and Cyber Security, and Machine Learning.

Nabeel Bani-Hani, received his Bachelor's and Master's degrees in Computer & Information Technology from the Institute of Information Technology - at the University of Sindh in 1998 and 2000 respectively. Currently, he is a Lecturer, at the University of Bahrain, College of Applied Studies.