Computer Science and Information Systems 00(0):0000–0000 https://doi.org/10.2298/CSIS123456789X

Sentence embedding approach using LSTM auto-encoder for discussion threads summarization

Abdul Wali Khan[1], Feras Al-Obeidat[2], Afsheen Khalid[1], Adnan Amin[1], and Fernando Moreira[3]

[1] Center for Excellence in Information Technology,Institute of Management Sciences
Peshawar, Pakistan
abdulwalikhanafridi@gmail.com,(adnan.amin, afsheen.khalid)@imsciences.edu.pk
[2] College of Technological Innovation, Zayed University
Abu Dhabi, UAE
Feras.Al-Obeidat@zu.ac.ae
[3] REMIT, IJP, Universidade Portucalense IEETA, Universidade de Aveiro
Portugal
fmoreira@uportu.pt

Abstract. Online discussion forums are repositories of valuable information where users interact and articulate their ideas, opinions, and share experiences about numerous topics. They are internet-based online communities where users can ask for help and find the solution to a problem. On online discussion forums, a new user becomes exhausted from reading the significant number of replies in a discussion. An automated discussion thread summarizing system (DTS) is necessary to create a candid view of the entire discussion of a query. Most of the previous approaches for automated DTS use the continuous bag of words (CBOW) model as a sentence embedding tool, which is poor at capturing the overall meaning of the sentence and is unable to grasp word dependency. To overcome this limitation, we introduce the LSTM Auto-encoder as a sentence embedding technique to improve the performance of DTS. The empirical result in the context of average precision, recall, and F-measure of the proposed approach with respect to ROGUE-1 and ROUGE-2 of two standard experimental datasets proves the effectiveness and efficiency of the proposed approach and outperforms the state-of-the-art CBOW model in sentence embedding tasks by boosting the performance of the automated DTS model.

Keywords: Sentence embedding, LSTM Auto-encoder, CBOW, Deep learning, Machine learning, NLP.

1. Introduction

Online discussion forums are web services where users can post a query about a specific topic and provide an online environment for individuals to articulate their thoughts. These online discussion forums are online communities where people with similar interests may exchange ideas, points of view, and experiences on a variety of topics. Because of user interaction and conversation, these forums become ideal archives of textual content. Online discussion forums may be used for various purposes, including getting students to discuss the course subject before class and reflecting on readings or assignments they have completed outside of class. Most of the queries generate huge replies in the discussion, so a new user becomes unable to scan all discussions and find the valuable and relevant text content shared by the users [1]. Some of the latest online discussion forums facilitate the users' finding their problem-relevant user discussions [2].

In our daily life, we frequently engage in multidimensional conversations with each other through blogs and online discussion forums, and online video meetings. The overwhelming amount of data generated from online interaction sometimes leads to information overload problems. On online discussion forums, the users hurriedly reply to the query, which may not always be relevant to the question. Extracting the relevant content from this growing raw text is a tough task for new users of the discussion forum [3]. This becomes stress-inducing and discourages the user's perception of online discussion forums. To maintain problem-relevant user replies in an online discussion forum, monitoring and filtering processes should be used. By using this process, new users will be able to easily find relevant content in thread discussions. The authors in the literature use numerous approaches to grasp the relevant and most valuable text content from this massive amount of data. Some of the commonly used methods are user-phrase queries to extract the most relevant text content from textual data [4], and an act-guided approach for tweets summarization based on word-based and symbol-based features [5], [6].

This study proposes an automated DTS model for online discussion forums that can automatically extract query-relevant user replies. The proposed model is based on the LSTM Auto-encoder techniques which is a deep learning architecture for sentence embedding to transform the sentences of the discussion replies into feature space for the extraction of the most relevant and significant text content using different similarity measures between the query and replies of the discussion. By using this automated model, users of online discussion forums can easily grasp the idea of the entire discussion by generating a candid view of the entire discussion.

The proposed technique for sentence embedding is a novel approach for embedding replies to sentences in online discussion forums. In literature, the CBOW model has been a widely used technique for sentence embedding in the field of natural language processing (NLP) for text summarizing tasks that have multiple flows for sentence embedding. This model considers only the surrounding words and ignores the structure and order of the words in the sentence. The CBOW model is also sensitive to the frequency of words, in which case common words have a high impact on sentence embedding. For the vector representation of a sentence, this model uses the average word vector approach, which disregards the word order in the long sequence of words [7]. We used the LSTM Auto-encoder in our model, which has the capability to remember patterns in long sequences of input. The model is fully generic and can be used for the summarizing of thread discussion of any English-based online discussion forum. The rest of the paper is organized as follows.

In Section 2, we review the previous state-of-the-art approaches for text summarization, which include numerous techniques related to text summarization, extractive and abstractive summarization approaches, and applications of discussion thread summarization. section 3 consists of the most important part of this study. The purpose of this section is to describe a novel approach to thread discussion summarization using the LSTM auto-encoder technique for automated summarization of thread discussion of online discussion forums. Section 4 elaborates on the outcomes, discusses the procedure that was suggested, and compares it with the most effective technique for the task of embedding sentences, section 5 describes the conclusion and future work of the study.

2. Literature Review

As an integral part of Natural Language Processing, text summarization has a wide range of applications, including news summarization, email thread summarization, social media content summarization, and thread discussion summarization of online discussion forums. There are two main approaches to text summarization which are extractive and abstractive summarization. In extractive summarization, the most significant textual chunks are extracted from the source text and merged in an extractive summary to reflect the core concept and flow of the original text. Abstractive summarization is a more complex and critical task that paraphrases the original text in a new version which can reflect the main clue of the original text. Based on the source documents, Text summarization is further divided into two classes which are multi-document and single-document summarization.

The extractive summarization approach for text summarization has various applications. The authors in the literature employ various techniques for the extraction of relevant text chunks. For an extractive summary generation, the text needs to be classified in order to identify and extract the relevant chunks. In the classification technique, similar textual units are classified in the same clusters independent of their significance in textual data [8]. For document categorization in some studies, the textual units are treated as typical sentences [9], in meeting conversations summarization the units are usually utterances [10], [11], and in the case of DTS, the units treat as a reply sentence [12],[13]. In the next step, various ranking methods are used to identify the importance of each textual chunk and assign a salience score to each textual chunk to arrange all in decreasing order based on their score. The textual chunks with a high score are considered to be the most significant. Based on a specific threshold or predefined cutoff, the significant textual chunks are extracted for the final summary. Cue dictionary techniques introduced in which the significance chunks of text are computed based on the presence or absence in the cue dictionary [14], In the title method, the weight of the sentences is computed based on the sum of all the text appearing in the title of the heading of source documents. For the extraction of relevant content from text [15] uses ontology and TF-IDF concept-based clustering approach for extractive summarization, similarly [16] uses numerous text mining techniques to create an extractive summary of the patent record. The TF-IDF and machine learning-based techniques are also used for the extraction [17].

One of the most critical parts of the extractive summarization process is to keep both coherence and consistency of the previously chosen textual units in the newly created version of the original text [18], [19]. Some feature-based summarization methods such as cue phrases and sentence locations are proposed to identify the relevant sentences for a final summary generation [20]. A combined TF-IDF and ontology tree structure techniques introduce for the extraction of keywords which is used for the selection of salient textual units from source text documents. after extraction and selection of keywords, a clustering technique is applied to cluster salient sentences to extract the relevant text chunks from a condensed text [21]. Search engine-based techniques are proposed on an extended query using the WordNet database to find the relevant web pages and on the bases of relevant keywords, the relevant identified sentences are extracted for final summarization [22].

Recently Graph-based approaches have been proposed in some studies for extractive summarization. Graph-based approaches are used as a PageRank (PR) algorithm [23] to rank the different textual units in sentences or passages. In graph-based approaches a sentence is represented as a node in a graph, if this node has a certain relation in a graph then it is more salient for the final summary [24].

Due to the advent of deep learning in the field of NLP, extractive summarization has become an exciting field of research. using deep learning for text-processing jobs, the performance of various machine learning methods improved. Recent research studies proposed many deep learning techniques for extractive summarization such as Query-oriented based extractive summarization proposed using the deep auto-encoder (AE) to compute the feature space from the input of term-frequency (TF) based on two vocabularies [25]. Similarly, [16] uses Tree Augmented NBN (TAN) variance of Bayesian network to generate extractive summarization of patent record. Sequence-to-sequence auto-encoder for extractive summarization is also used for long and noisy social media text content [26], Multi-document extractive text summarization approach is proposed which uses an auto-encoder neural network to compare the scoring and performance of multiple documents [27].

In this study, we use LSTM Auto-encoder for sentence embedding and an extractive approach for DTS, similar to the one suggested by [27], which is based on recurrent neural network architecture. The proposed LSTM Auto-encoder of this study is also a deep learning architecture that is mostly used as embedding techniques for image data but recent research studies prove that it is also a powerful tool for text data [7]. LSTM auto-encoder consists of two parts which are Encoder and Decoder. The input sequence is read by the encoder which encodes the entire input sequence into an internal representation. The second part decoder reads the internal representation of the encoder and generates the output sequence.

2.1. Preliminary study

LSTM Auto-Encoder as a sentence embedding: The sentence embedding vector can be obtained by using the average word vector in a sentence. Similarly, we can obtain the paragraph vector by calculating the average sentence vectors in a paragraph. The average vector technique is inefficient in capturing the semantic information in sentences. To obtain the rich embedding vector of the sentence we used LSTM Auto-encoder to grasp the semantic relation between words in a sequence of input using the recurrent neural network architecture for the DTS model.

Auto Encoder: Previous approaches for automated DTS use different embedding techniques. One of the most common techniques is the Continues Bag of Words (CBOW) model [2]. All these embedding techniques are based on the distributional hypothesis in which the similarity of words is based on their contextual representation in the sentence [7]. With the emergence of deep neural networks in the field of Natural language processing, neural word embedding received a lot of attention [28]. Many research studies prove that the neural word embedding technique is a powerful tool for understanding the semantic relation between words such as dynamic convolutional neural network (DCNN) for sentence embedding tasks to extract the most active feature from sentences [29]. This model allowed extracting the most active features in the sentences independently of their location. The authors of [30] build a general-purpose sentence encoder for multiple objectives which are classification of text, machine transition, parse tree generation, and skip-thought tasks. The proposed model is trained by several data sources on over 100 million sentences with multiple training objectives. In [31] the authors proposed a recursive auto-encoder (RAE) based on unfolding objectives for the similarity of two sentences. In [32] the authors proposed an auto-encoder model which can denoise the sentence by deleting words and changing the positions of words in a sentence where the decoder is used to reconstruct the original sentence by swapping bi-grams. In [33] the authors introduce a neural topic model to capture the global semantic meaning of the document and integrate that model with an automatic text summarization model. [34] Proposed a hierarchical encoder-decoder model for DTS which is pre-trained on synthetic interleaved text. This approach aims to overcome the limitation of the traditional thread discussion summarization system.

Long Short-Term Memory (LSTM): LSTM is another variant of recurrent neural networks. An LSTM-based neural network is better than traditional neural networks due to its memory. The traditional neural network is not virtuous at memorizing short-term patterns and also suffers from vanishing gradient problems [35]. The LSTM improves the performance of the traditional neural network in these problems. Numerous studies use LSTM in many natural language processing tasks such as a combined model based on word embedding using LSTM for semantic similarity in text classification tasks [36]. LSTM-based sentence encoder that is trained by an annotated training corpus to capture the useful feature from sentences and use these features for text classification tasks [37]. LSTM-CNN proposed by [38] for extractive summarization to construct new sentences by exploring semantic phrases of text. A multilayered attentional peephole convolutional long short-term memory (LSTM) for extractive text summarization task [39]. The model is based on an attentional mechanism that gives weight to the most significant parts of the text. Attention-based bidirectional LSTM is used which is also known as BiLSTM for text generation to enhance the correlation between generated text and the source text [40]. This model is also able to eliminate repeated words and solves out-of-vocabulary word problems. The use of LSTM in Auto-encoder is demonstrated in Fig. 1 improves the performance of automated thread discussion summarization. Text processing tasks are highly dependent on the representation of the text in the feature space. Most of the machine learning or deep learning algorithm performs better using efficient embedding techniques [41].

Text quality features: Text features refer to words or groups of words in a text which help to understand the core idea of the text. In text processing, text features play very important roles. Some of the most common text features are as follows:

Semantic distance between texts: Word Mover's Distance (WMD) is the most commonly used technique for semantic distance calculation between text documents. This technique is used to find the minimum cumulative distance between two text documents in multi-dimensional space.

$$Sem.Distance = WMD\ (text\ A,\ text\ B) \tag{1}$$

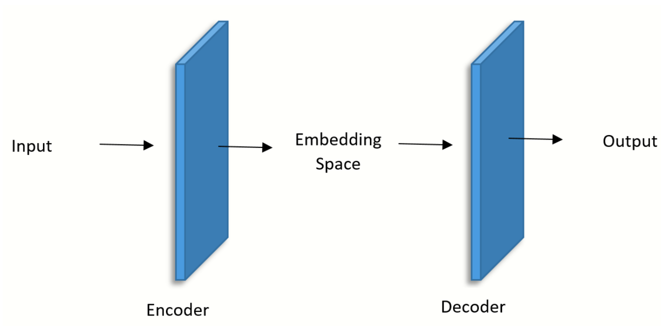

Fig. 1. LSTM Auto-encoder based sentence embedding

Cosine similarity between texts: Cosine similarity is the process of finding the similarity between the inner product of the vectors of two texts. It calculates the angle between the vectors of the text documents. In Python, cosine similarity can be calculated using the following formula.

$$\mathit{Cosin.Sim}=\mathit{Cosine\_Sim}(\mathit{text\ A},\mathit{text\ B}) \tag{2}$$

Unique word count: Unique words are unrepeated words in the text which are considered text features in text processing. It can be found by counting the unique words minus the repeated words in a sentence or text. In Python, the following formula can be used to count the unique words.

$$\mathit{Uniqe\_Word}=\mathit{Unique\_words}(\mathit{text}) \tag{3}$$

Common overlapping words in texts: overlapping words are those words that have a common semantic characteristic. These words can be found by using Jaccard similarity in Python using the following formula which finds the similarity of asymmetric binary vectors of text A and B.

$$\mathit{Com\_words}=\mathit{Jaccard\_Sim}\ (\mathit{text\ A},\ \mathit{text\ B}) \tag{4}$$

Length of texts: text length is another feature of text in which the length of two texts or two sentences is used as a common feature. In python the following formula can be used to find the length of a text.

$$Text\_length = \frac{\mathit{No.\ of\ words\ in\ a\ text}}{\mathit{Max\ length\ of\ text}} \tag{5}$$

Number of Nouns and Verbs in a text: In this technique, the number of nouns and verbs are considered as text features. In python, the following formula is used to find the number of nouns and verbs in texts.

$$Noun\_verb = \frac{\mathit{No.\ of\ verbs\ and\ nouns}}{\mathit{Text\ length}} \tag{6}$$

3. Methodology

In this section, we discuss the proposed methodology to summarize the discussions of online discussion forums using an automated discussion thread summarization model. The order of the steps is shown in Fig. 2 and a description of each step is given below.

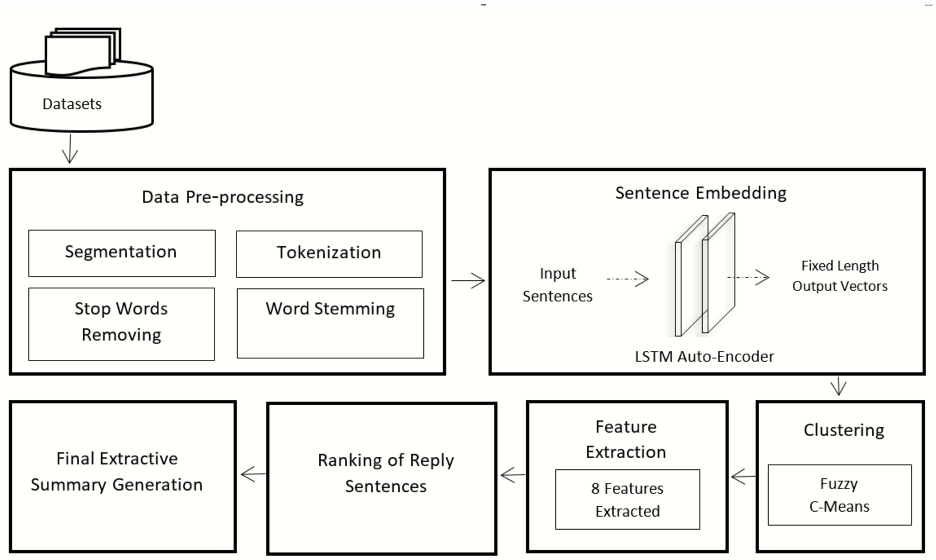

Fig. 2. Methodology Framework

3.1. Datasets

In this study, we used two standard discussion forum datasets: Ubuntu datasets generated by Ubuntu Online discussion forums and *NYC* datasets obtained from the TripAdvisor discussion forum. The Ubuntu dataset contains *756* user conversations. Similarly, the *NYC* dataset has *788* user conversations. for evaluation of the proposed methodology, the query of each thread is referred to as the initial posts and the replies to the query are called the candidate answers.

3.2. Data preprocessing

The preparation of data is the first step for any machine-learning task. To prepare data for our machine learning algorithm, we performed the following four types of text preprocessing tasks.

1. *Sentence segmentation:* This is the process of splitting lengthy text into small chunks or sentences. Sentence boundaries are the points where a text sequence is split into sentences. We have used signs of interrogation (?), exclamation (!), and full stop (.) as sentence boundaries to segment the text.

2. *Tokenization:* After segmenting the text into sentences, word tokenization is performed. Different techniques have been applied to split the sentence into tokens of words such as the white space technique, we have divided the reply sentences into tokens of words using the widely used tools for text processing which is *(NLTK)* library of Python.
3. *Removing stop words:* Stop words such as “the”, “an”, “a”, “is”, “all”, etc. are not significant words. These words are used frequently in the text and cannot be identified as having any particular value, hence they are not taken into account in our feature-embedded space. To reduce noise and prepare the text for the machine learning model to be applied, these words need to be removed.
4. *Word stemming:* It is the process in which each word is derived into its base or root word. The word-stemming process aims to help the algorithm to catch the word similarity in embedding. An example of word-stemming is to convert the stem words *‘playing’, ‘played’, ‘plays’* to its root word which is *‘play’*. In this process, we used the Porter stemmer to get the stems of the words.

3.3. Embedding of reply sentences

Word embedding is the building block of sentence embedding that transforms the distinct words into a single vector of 1s and 0s. This type of embedding is known as one-hot encoding and is completely dependent upon the corpus. In the vector of this embedding technique, 1’s indicates the presence of a word and 0’s represents the absence of a word in the corpus. In this approach, we cannot capture any semantic information in a sentence.

Sentence embedding is the extension of word embedding where the entire variable-length sentence is converted into a fixed numerical vector. One of the simplest ways of converting a variable-length sentence into a fixed-length vector is to encode all the words of the sentence into vectors and then take the average of all these word vectors in a single numerical vector. This average word vector approach is followed by Word2vec models, which is not sufficient for capturing various semantic information, such as word ordering information and other semantic relation between the words. In textual data Transformation tasks, most of the previous approaches used averaging vector concepts [2] or weighted representation of text using TF-IDF [21]. In the literature, the word2vect is used in two versions: the CBOW model and the skip-gram model. The CBOW approach predicts the word using the surrounding context while the skip-gram model is the inverse of CBOW which uses the distributed representation of the input word and predicts the context. to capture the semantic information of a sentence, word2vect model embedding is unsuitable as they provide semantic information only in a limited context.

LSTM Auto-encoder is a recurrent neural network architecture demonstrated in Fig. 3. With the help of using LSTM, Auto-encoder can memorize the previous sequence of words and can capture word order dependency and other semantic information in a long sequence of words and produce a rich embedding vector of a sentence. We used LSTM encoder and decoder models and a combined model for the sentence embedding process. The first encoder model takes the tokenized sequence and embeds it into dense vectors of fixed length. the embedded vectors are then fed into an LSTM encoder layer with 128 units to learn the encoding of the input sequence into a fixed-length vector representation. In the second step, the decoder model takes the fixed length vector from the encoder model and generates the output sequence. the decoder model also has an LSTM layer with 128

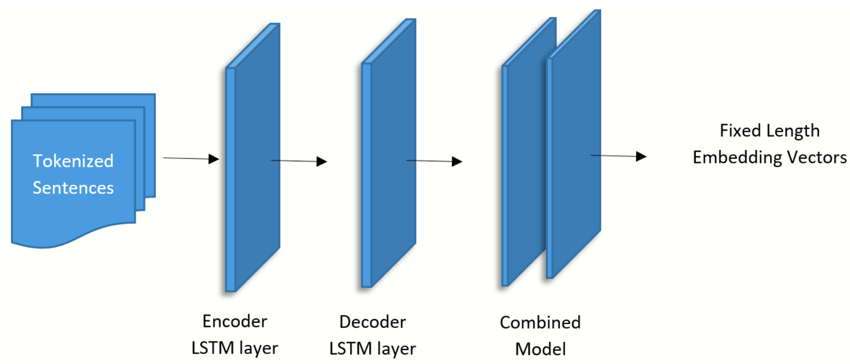

Fig. 3. LSTM Auto-encoder based sentence embedding.

units, followed by a dense output layer that produces a probability distribution over the output tokens. the last model is the combined model which integrates the encoder and decoder models. The purpose of the combined model is to learn the fixed-length vector representation of the input sequence and minimize the loss. we trained our model for *10* epochs. during training the model learn to generate a fixed-length vector representation of the input sequence.

The proposed study also uses the CBOW model for the assessment of the proposed sentence embedding technique. Both LSTM Auto-encoder and CBOW models are applied on both datasets for sentence embedding tasks. After applying embedding approaches, three clustering techniques are applied respectively on each dataset which is briefly described below.

3.4. Clustering

In thread discussions of online discussion forums, different users participate and share opinions about the topic. The initial question or query received mostly semantic similar replies. For extraction of the most relevant user replies to the initial question, similar replies must be clustered together to extract the most relevant replies from the discussion. In clustering, the most similar text chunks are clustered together is the crucial phase of clustering, from which Nemours scoring procedures are used to retrieve largely pertinent replies. Clustering has two types which are hard clustering and soft clustering. In hard clustering, each data point belongs to only one cluster, and in the soft clustering approach, the data point may belong to many clusters. In soft clustering, a similarity score is used which is also known as the membership score of each data point which designates the significance of the data points towards the cluster centroid which is an average vector of all the clustered sentence vectors.

In the proposed methodology we used three clustering techniques to cluster the reply sentences which are K-means, K-medoid, and FCM. Out of these three techniques, FCM outperforms other clustering techniques with LSTM Auto-encoder-based sentence embedding vectors. FCM is a soft clustering technique that assigns a likelihood score to each reply sentence [42]. The way FCM works Initially, a sentence may belong to more than one cluster which converges to only one cluster after FCM iterations. The iterations use a step-wise approach to converge a sentence to only one cluster. These steps recalculate the centroid of the cluster and the membership score of the reply sentence [43].

Fuzzy C-means convergence

1. Selected the number of clusters K = 10
2. Initially random values are assigned to each sentence which shows the probability of a sentence to clusters where pi is the point and k is the cluster (pi, k).

$$\mu_k(n+1) = \frac{\sum_{p_i \in k} P_i * P(\mu_k|P_i)^b}{\sum_{p_i \in k} P(\mu_k|P_i)^b} \tag{7}$$

3. After random initialization, iteratively cluster centroid and membership score are recalculated until the convergence.
4. The iteration will be continued until convergence or until the user-specified limit of iteration.

After the clustering process, We extracted only a single sentence from each cluster based on a certain score assigned to each data sentence in the clustering using quality text features which are discussed in the next step.

3.5. Quality text features extraction

In the extractive text summarization process mostly text features are used as a scoring technique. The purpose of text feature extraction is to identify the salient sentence in the user reply sentences based on the assigned text feature score. Based on these quality features each sentence is scored in all clusters. A high score for the sentence indicates that the particular sentence has all the quality features and it is the most relevant one to the query of the discussion. In this work, eight different types of quality text features are extracted. The feature values for each reply sentence are normalized between zero and one which are further described below.

1. *Semantic distance between thread reply and thread centroid:* Semantic distance means the semantic difference between replies and thread centroid. TF-IDF is used to calculate the thread centroid. The thread centroid is the centre point of all replies to thread discussions. After the mapping of the thread centroid vector, word mover distance (WMD) is used to calculate the semantic distance between the reply sentence vector and the thread centroid vector in each thread discussion. In this process, the distance between the thread reply and the thread centroid is calculated. This technique extracts the most important and unique terms/words in a thread which is also known as the features of that thread discussion.
2. *Cosine similarity between reply sentences and thread centroid:* In cosine similarity, the cosine angle is calculated between the reply sentence vector and the thread centroid vector. Cosine similarity is a multi-dimensional space technique used for the measurement of the similarity between two vectors. The purpose of this step of feature extraction is to capture the cosine similarity between reply sentence vectors and thread centroid vectors as text features.

3. *Unique word count in a reply sentence:* In this step of feature extraction, the unique word in each reply sentence is counted. This unique word played a very important role as a feature of the reply sentence. A reply sentence is considered for a final summary generation if it contains unique words.
4. *Common or overlapping words between thread reply and initial post:* Common or overlapping words are those words that share common semantic characteristics. In this step of feature extraction, the overlapping word between the reply sentence and the initial post is extracted. This task is performed using Jaccard similarity which is used for the similarity of asymmetric binary vectors of thread reply sentences and initial posts or queries.
5. *Semantic Distance between thread reply sentence and thread title:* Semantic similarity is also an important feature of extractive summarization. In this step, the semantic similarity between the reply sentence of a discussion and the thread title is calculated. For this purpose, the word mover distance (WMD) is used to calculate the semantic distance between reply sentences and thread titles.
6. *Semantic Distance between thread reply sentences and initial post:* A reply sentence is considered to be salient for the final summary if it has semantic similarity with the initial post. Word mover distance (WMD) is used for the calculation of semantic distance between thread reply sentences and initial posts.
7. *Length of reply sentence:* Reply sentence length means the number of words in a sentence. In this step, the number of words in each sentence is counted as a feature of a reply sentence to a discussion.
8. *Number of Nouns and Verbs in a reply sentence:* In this step, the number of verbs and nouns in a thread reply sentence is counted. The number of verbs and nouns considered is a feature of thread reply sentences.

3.6. Ranking of Reply Sentences

In this phase, a ranking score is assigned to each reply sentence in each cluster. This score indicates the salient sentences for extraction from thread discussions and helps to eliminate irrelevant replies for the final summary generation. In the previous step, different quality text features were extracted from reply sentences. In the ranking process, the extracted features are used as a scoring technique for each sentence. A sentence that has a high score is considered to be a salient sentence for a final summary of the thread discussion. According to the extracted features, if a sentence has all these features, it will be more analogous to the asked query. Based on eight quality text features, each sentence is represented as an eight-dimensional vector that specifies the significance of a sentence based on a certain distance from centroid vectors of the cluster. To score each sentence, the text features score is calculated for each reply sentence in each cluster. After individual calculation, all features are summed up to score each reply in clusters. The summation function of features is as follows;

$$Score(sentence) = \sum_{k=1}^{8} reply\_sent_{f_i} \tag{8}$$

Where *Score(sentence)* indicates the overall score of reply sentences and *reply_sent* represents the features score of each reply sentence. After the calculation of feature scores, each sentence in each cluster is ranked based on this feature score. As in the previous section of clustering, the number of clusters is 10. In the next step, only a single sentence is extracted based on these features.

3.7. Summary Generation

The most important part of extractive summarization is to take the most relevant sentence from the source text and place it in order to maintain the overall concept of a paragraph. Numerous techniques are used in the literature for extraction to obtain relevant replies for the final summary which are discussed in the literature review section. To extract the most relevant replies to the initial post, we proposed eight different types of text features in this study. Before extraction of relevant replies from discussions all discussion replies are clustered in 10 clusters using FCM clustering algorithms. The purpose of clustering is to group similar sentences and apply text feature extraction techniques to identify the most relevant replies for extraction. based on these features, only one high-ranked sentence is extracted for a final summary generation.

4. Result and Discussion

In this section, we elaborate on the effectiveness of our proposed sentence embedding technique and compare it with the state-of-the-art CBOW embedding model in the context of two standard discussion forums Ubuntu and TripAdvisor datasets. The average recall, precision, and F-measure obtained with ROUGE-N (N = 1, 2) [44] are used to compare the effectiveness of our proposed sentence embedding technique with the alternative CBOW embedding model. The empirical result of ROUGE-1 using two discussion forums datasets shown in Tables 1 and 2 illustrates the performance of the proposed sentence embedding technique in comparison of FCM, K-Medoid and K-means clustering algorithms the FCM outperform other clustering algorithms using the proposed sentence embedding technique.

Table 1. ROUGE-1 of NYC dataset based using both LSTM Auto-encoder and CBOW embedding models

Average Metrics	Algorithms	LSTM Auto-encoder	CBOW model
Precision	Kmediod	24.80	25.70
	FCM	39.43	26.95
	Kmeans	26.78	21.45
Recall	Kmediod	34.27	35.33
	FCM	40.60	24.46
	Kmeans	36.58	26.79
F measure	Kmediod	28.25	29.32
	FCM	39.86	24.39
	Kmeans	30.33	23.14

Table 2. ROUGE-1 of Ubuntu dataset based on both LSTM Auto-encoder and CBOW embedding models

Average Metrics	Algorithms	LSTM Auto-encoder	CBOW model
Precision	Kmediod	34.92	35.86
	FCM	37.11	31.09
	Kmeans	36.20	33.10
Recall	Kmediod	36.79	37.63
	FCM	39.33	29.73
	Kmeans	38.88	35.28
F measure	Kmediod	35.63	36.53
	FCM	38.08	29.79
	Kmeans	37.10	33.94

It is clear from the figures 4 and 5 in the context of average precision, recall, and F-measure of ROUGE-1 using two standard datasets, the proposed sentence embedding technique outperforms the CBOW embedding model in terms of average precision, recall and F-measure.

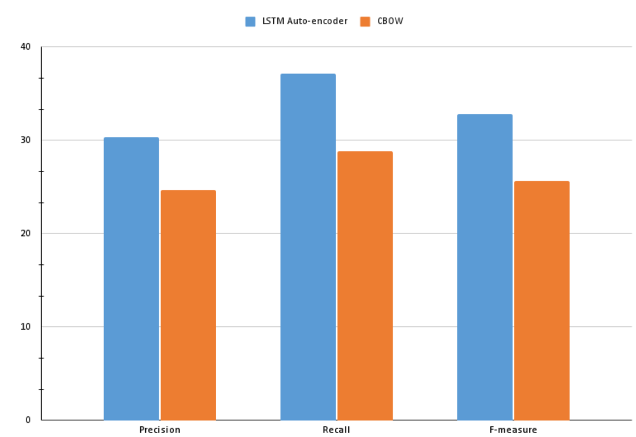

Fig. 4. ROUGE-1 of LSTM Auto-encoder and CBOW model using NYC dataset.

Referring to the ROUGE-2 result presented in Table 3 and 4 The proposed sentence embedding technique performs better than the CBOW embedding model. In comparison to the three clustering algorithms FCM, K-Mediod and K-means, in terms of two standard discussion forums datasets FCM offered better outcomes for summarization using the embedding vectors of the proposed sentence embedding technique.

In the context of average precision, recall and F-measure of ROUGE-2, illustrated in the figure 6 and 7, the proposed sentence embedding technique outperforms the CBOW embedding models in sentence embedding tasks for text summarization. Using the pro-

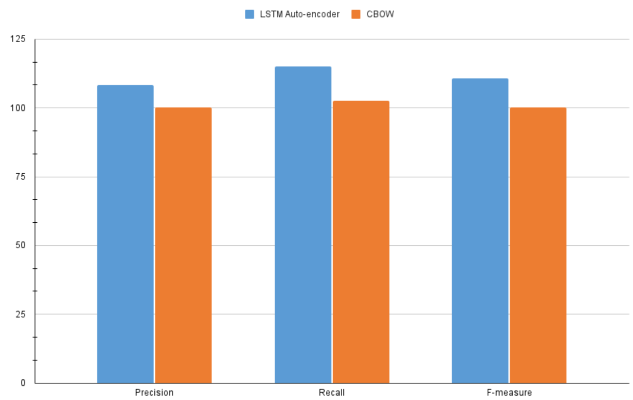

Fig. 5. ROUGE-1 of LSTM Auto-encoder and CBOW model using Ubuntu dataset.

Table 3. ROUGE-2 of NYC dataset based on both LSTM Auto-encoder and CBOW embedding models

Average Metrics	Algorithms	LSTM Auto-encoder	CBOW model
Precision	Kmediod	5.14	6.79
	FCM	16.87	8.31
	Kmeans	7.47	3.68
Recall	Kmediod	7.81	10.04
	FCM	17.59	6.81
	Kmeans	10.73	5.29
F measure	Kmediod	6.05	7.93
	FCM	17.14	7.12
	Kmeans	8.60	4.13

Table 4. ROUGE-2 of Ubuntu dataset based on both LSTM Auto-encoder and CBOW embedding models

Average Metrics	Algorithms	LSTM Auto-encoder	CBOW model
Precision	Kmediod	11.27	13.98
	FCM	15.33	8.50
	Kmeans	14.74	12.41
Recall	Kmediod	11.65	14.49
	FCM	16.63	8.52
	Kmeans	16.38	12.60
F measure	Kmediod	11.44	14.19
	FCM	15.89	8.27
	Kmeans	15.32	12.47

posed sentence embedding technique, the results of the experiment indicate that the automated summarization model performs better on a dataset taken from Ubuntu discussion forums as well as a dataset taken from NYC discussion forums when assessed.

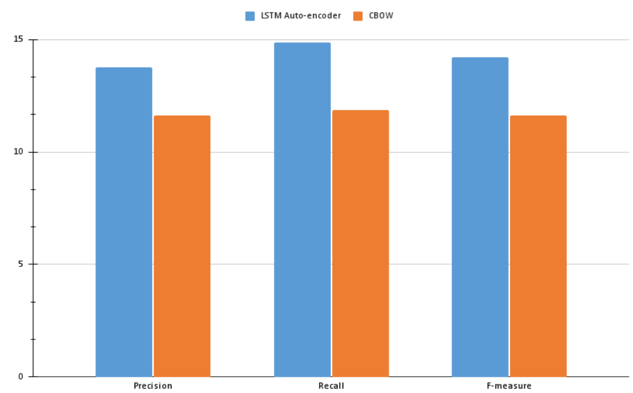

Fig. 6. ROUGE-1 of LSTM Auto-encoder and CBOW model using Ubuntu dataset.

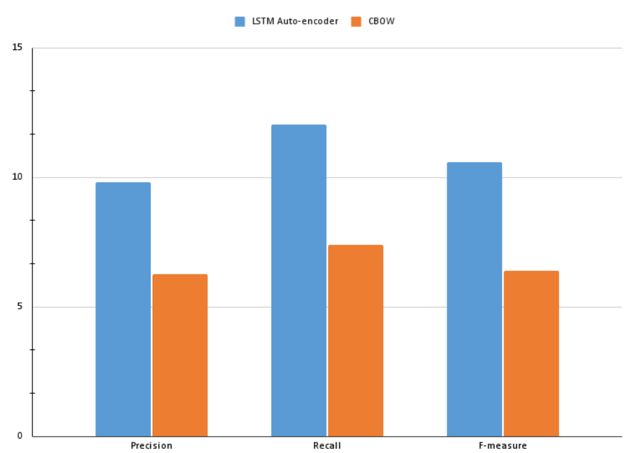

Fig. 7. ROUGE-1 of LSTM Auto-encoder and CBOW model using NYC dataset.

4.1. Comparison

By comparing our proposed technique with the state-of-the-art CBOW model which uses the average word vector technique for sentence embedding. The empirical result given in Table 5 and graphical visualization in Figure 8 proved that our proposed sentence embedding technique performs better than the CBOW approach with FCM clustering.

Table 5. Comparison with state of the art approach [2]

Average Metrics	Algorithms	LSTM Auto-encoder	CBOW model
Precision	Kmediod	24.8	34.68
	FCM	39.43	28.2
Recall	Kmediod	34.27	39.83
	FCM	40.6	28.31
F measure	Kmediod	28.25	36.03
	FCM	39.86	27.46

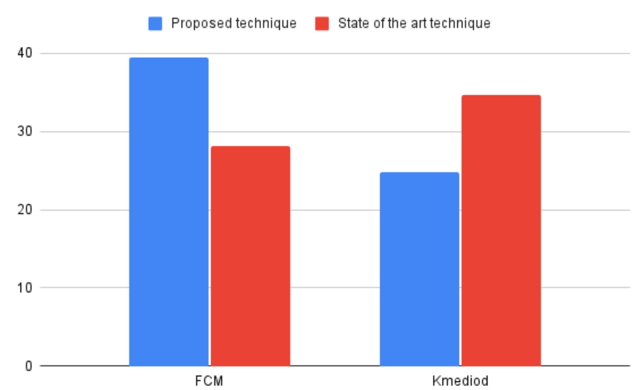

Fig. 8. Comparison with state of the art approach [2]

5. Threat to validity

Selection bias: The proposed LSTM auto-encoder method was examined on only two standard datasets, which may not be typical of all conceivable datasets.

Generalizability: The study only evaluated the proposed approach on DTS tasks, and it is unclear whether the proposed approach would perform similarly on other NLP tasks or in different domains. Therefore, the generalizability of the proposed approach to other tasks or domains is uncertain.

Evaluation metric bias: The proposed method was evaluated using the ROUGE-1 and ROUGE-2 assessment metrics. Although these measures are commonly employed in text summarization tasks, they may not capture all aspects of summary quality, and alternative metrics may provide a different view of the performance of the proposed method.

6. Conclusion and future work

Thread discussion summarization is a challenging task in the field of NLP. The recent trend of deep learning-based natural language processing has proposed new techniques that attract the researchers' attention. This study introduces a deep learning-based sentence embedding approach using a recurrent neural network architecture to boost the performance of automated DTS. The state-of-the-art CBOW model is an inefficient approach for sentence embedding and is unable to capture semantic information of the overall sentence due to its commutative calculation of embedding vectors. To overcome the limitations of the CBOW model in DTS, this study proposed an LSTM Auto-encoder for efficient sentence representation in embedding space. The proposed methodology is evaluated on two standard datasets and compares both the CBOW model and LSTM Auto-encoder in sentence embedding tasks. In the context of precision, recall, and F-measure, the empirical results prove that the LSTM Auto-encoder improves the performance of the DTS model concerning both ROUGE-1 and ROUGE-2 evaluation metrics.

In the future, we plan to modify and apply our proposed methodology to the non-English online discussion forums dataset. Deep learning-based clustering approaches will also be considered to further enhance the performance of the proposed methodology. Furthermore, we plan to extend our proposed approach for the abstractive summarization tasks and also intend to compare our novel sentence embedding technique with other recently introduced neural sentence embedding techniques [28].

7. Author contributions

A novel RNN-based technique is proposed by the authors for sentence embedding. The empirical comparison between the proposed method and the CBOW model demonstrates the significance of the novel approach for sentence embedding and improves the performance of the automated DTS model for online discussion forums.

8. Acknowledgements

This work was supported by the FCT – Fundação para a Ciência e a Tecnologia, I.P. [Project UIDB/05105/2020

References

1. Liang Zhou and Eduard H Hovy. On the summarization of dynamically introduced information: Online discussions and blogs. In *AAAI Spring symposium: Computational approaches to analyzing weblogs*, page 237, 2006.

2. Atif Khan, Qaiser Shah, M Irfan Uddin, Fasee Ullah, Abdullah Alharbi, Hashem Alyami, and Muhammad Adnan Gul. Sentence embedding based semantic clustering approach for discussion thread summarization. *Complexity*, 2020, 2020.
3. Akram Osman, Naomie Salim, and Faisal Saeed. Quality dimensions features for identifying high-quality user replies in text forum threads using classification methods. *PloS one*, 14(5):e0215516, 2019.
4. Yashar Mehdad, Giuseppe Carenini, and Raymond Ng. Abstractive summarization of spoken and written conversations based on phrasal queries. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1220–1230, 2014.
5. Renxian Zhang, Wenjie Li, Dehong Gao, and You Ouyang. Automatic twitter topic summarization with speech acts. *IEEE transactions on audio, speech, and language processing*, 21(3):649–658, 2012.
6. David M Zajic, Bonnie J Dorr, and Jimmy Lin. Single-document and multi-document summarization techniques for email threads using sentence compression. *Information Processing & Management*, 44(4):1600–1610, 2008.
7. Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *arXiv preprint arXiv:1608.04207*, 2016.
8. Julian Kupiec, Jan Pedersen, and Francine Chen. A trainable document summarizer. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 68–73, 1995.
9. Feifan Liu and Yang Liu. Correlation between rouge and human evaluation of extractive meeting summaries. In *Proceedings of ACL-08: HLT, short papers*, pages 201–204, 2008.
10. Matthew Marge, Satanjeev Banerjee, and Alexander Rudnicky. Using the amazon mechanical turk to transcribe and annotate meeting speech for extractive summarization. In *Proceedings of the NAACL HLT 2010 workshop on creating speech and language data with Amazon's Mechanical Turk*, pages 99–107, 2010.
11. G van Oortmerssen, S Raaijmakers, M Sappelli, E Boertjes, S Verberne, N Walasek, and W Kraaij. Analyzing cancer forum discussions with text mining. 2017.
12. Ruben Sipos, Pannaga Shivaswamy, and Thorsten Joachims. Large-margin learning of submodular summarization models. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 224–233, 2012.
13. Carlos N Silla Jr, Celso AA Kaestner, and Alex A Freitas. A non-linear topic detection method for text summarization using wordnet. In *Proceedings of the Workshop of Technology Information Language Human (TIL 2003)*, 2003.
14. Gabriel Murray, Steve Renals, and Jean Carletta. Extractive summarization of meeting recordings. 2005.
15. Amy JC Trappey, Charles V Trappey, and Chun-Yi Wu. Automatic patent document summarization for collaborative knowledge systems and services. *Journal of Systems Science and Systems Engineering*, 18(1):71–94, 2009.
16. Yuen-Hsien Tseng, Yeong-Ming Wang, Yu-I Lin, Chi-Jen Lin, and Dai-Wei Juang. Patent surrogate extraction and evaluation in the context of patent mapping. *Journal of Information Science*, 33(6):718–736, 2007.
17. Sanda M Harabagiu and Finley Lacatusu. Generating single and multi-document summaries with gistexter. In *Document Understanding Conferences*, pages 11–12. Citeseer, 2002.
18. Vishal Gupta and Gurpreet Singh Lehal. A survey of text summarization extractive techniques. *Journal of emerging technologies in web intelligence*, 2(3):258–268, 2010.
19. Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

20. Denis Vazhenin, Satoru Ishikawa, and Vitaly Klyuev. A user-oriented web retrieval summarization tool. In *2009 Second International Conference on Advances in Human-Oriented and Personalized Mechanisms, Technologies, and Services*, pages 73–78. IEEE, 2009.
21. Rahim Khan, Yurong Qian, and Sajid Naeem. Extractive based text summarization using k-means and tf-idf. *International Journal of Information Engineering & Electronic Business*, 11(3), 2019.
22. Jagadish S Kallimani, KG Srinivasa, and B Eswara Reddy. Summarizing news paper articles: experiments with ontology-based, customized, extractive text summary and word scoring. *Cybernetics and Information Technologies*, 12(2):34–50, 2012.
23. Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.
24. Gunes Erkan and Dragomir Radev. Lexpagerank: Prestige in multi-document text summarization. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 365–371, 2004.
25. Mahmood Yousefi-Azar and Len Hamey. Text summarization using unsupervised deep learning. *Expert Systems with Applications*, 68:93–105, 2017.
26. Shuming Ma, Xu Sun, Junyang Lin, and Houfeng Wang. Autoencoder as assistant supervisor: Improving text representation for chinese social media text summarization. *arXiv preprint arXiv:1805.04869*, 2018.
27. Afsaneh Rezaei, Sina Dami, and Parisa Daneshjoo. Multi-document extractive text summarization via deep learning approach. In *2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI)*, pages 680–685. IEEE, 2019.
28. Kathrin Blagec, Hong Xu, Asan Agibetov, and Matthias Samwald. Neural sentence embedding models for semantic similarity estimation in the biomedical domain. *BMC bioinformatics*, 20(1):1–10, 2019.
29. Richard Socher, Eric Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. *Advances in neural information processing systems*, 24, 2011.
30. Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.
31. Jeenu Grover and Pabitra Mitra. Sentence alignment using unfolding recursive autoencoders. In *Proceedings of the 10th Workshop on Building and Using Comparable Corpora*, pages 16–20, 2017.
32. Felix Hill, Kyunghyun Cho, and Anna Korhonen. Learning distributed representations of sentences from unlabelled data. *arXiv preprint arXiv:1602.03483*, 2016.
33. Thong Nguyen, Anh Tuan Luu, Truc Lu, and Tho Quan. Enriching and controlling global semantics for text summarization. *arXiv preprint arXiv:2109.10616*, 2021.
34. Sanjeev Kumar Karn, Francine Chen, Yan-Ying Chen, Ulli Waltinger, and Hinrich Schütze. Few-shot learning of an interleaved text summarization model by pretraining with synthetic data. *arXiv preprint arXiv:2103.05131*, 2021.
35. John Macintyre, Lazaros Iliadis, Ilias Maglogiannis, and Chrisina Jayne. *Engineering Applications of Neural Networks: 20th International Conference, EANN 2019, Xersonisos, Crete, Greece, May 24-26, 2019, Proceedings*, volume 1000. Springer, 2019.
36. Jenq-Haur Wang, Ting-Wei Liu, Xiong Luo, and Long Wang. An lstm approach to short text sentiment classification with word embeddings. In *Proceedings of the 30th conference on computational linguistics and speech processing (ROCLING 2018)*, pages 214–223, 2018.
37. Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. Learning general purpose distributed sentence representations via large scale multi-task learning. *arXiv preprint arXiv:1804.00079*, 2018.
38. Shengli Song, Haitao Huang, and Tongxiao Ruan. Abstractive text summarization using lstm-cnn based deep learning. *Multimedia Tools and Applications*, 78(1):857–875, 2019.

39. Md Motiur Rahman and Fazlul Hasan Siddiqui. Multi-layered attentional peephole convolutional lstm for abstractive text summarization. *ETRI Journal*, 43(2):288–298, 2021.

40. Jiawen Jiang, Haiyang Zhang, Chenxu Dai, Qingjuan Zhao, Hao Feng, Zhanlin Ji, and Ivan Ganchev. Enhancements of attention-based bidirectional lstm for hybrid automatic text summarization. *IEEE Access*, 9:123660–123671, 2021.

41. Erxue Min, Xifeng Guo, Qiang Liu, Gen Zhang, Jianjing Cui, and Jun Long. A survey of clustering with deep learning: From the perspective of network architecture. *IEEE Access*, 6:39501–39514, 2018.

42. Soumi Ghosh and Sanjay Kumar Dubey. Comparative analysis of k-means and fuzzy c-means algorithms. *International Journal of Advanced Computer Science and Applications*, 4(4), 2013.

43. Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

44. Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

Abdul Wali Khan pursued his academic journey with a passion for cutting-edge technologies. After completing his MSc from Peshawar University in 2017, he set his sights on delving deeper into the realm of artificial intelligence. Enrolling at the Institute of Management Sciences Peshawar, he dedicated himself to mastering the intricacies of machine learning, natural language processing, text mining, and big data. With unwavering determination, he successfully obtained his MS Degree in 2022, solidifying his expertise in this rapidly evolving field. With an eye toward the future, he is now in pursuance of his PhD, driven by his unwavering curiosity and commitment to making groundbreaking advancements in the world of AI.

Feras Al-Obeidat is an Associate Professor at the College of Technological Innovation at Zayed University. Dr. Al-Obeidat also holds an admin position as a Graduate Program Coordinator. He received his Master's and Ph.D. in Computer Science from the University of New Brunswick, Canada. Dr. Al-Obeidat's primary field of research in Artificial Intelligence and Machine Learning. Directly following his Ph.D., Dr. Al-Obeidat contributed to industrial, university, and government teaching and research with premier organizations including IBM Canada, the University of New Brunswick, Salesforce, and the National Research Council of Canada. Dr. Al-Obeidat is honored to join Zayed University and pursue and sustain ties with government and industry research.

Afsheen Khalid received Ph.D. in computational linguistics in 2018 from the Institute of Management Sciences, Peshawar, Pakistan. She completed her M.S. degree in computer engineering from CASE (Center of Advanced Studies in Engineering) in Islamabad, Pakistan in 2008. She is currently an Assistant Professor at the Department of Computer Science, Institute of Management Sciences. Her research interests include text mining, big data, NLP, and machine learning.

1 

Adnan Amin earned his M.Sc. in Computer Science from the University of Peshawar in 2008. He received his MS-CS degree (highest 1st class honors with distinction) and a Ph.D. degree (with Scholastic Honours) in data mining and machine learning from the Institute of Management Sciences (IMSciences) Peshawar, Pakistan, in 2015 and 2023, respectively. In November 2015, he joined IMSciences as a lecturer at the Center for Excellence in Information Technology. He is the lead of IM—DigiSol and the faculty adviser for the Computing and Innovation Society (IMCIS) at IMSciences. Before, he worked as an international consultant for curriculum and academic development at the School of ICT, NIMA Kabul, which was connected to the University of Jyvaskyla in Finland. This was part of a World Bank-funded project for Maxwell Stamp PLC, London (Project Code: P102573). Dr. Adnan's research and innovation interests are industry-driven and cross-disciplinary. He focuses on developing and commercializing digital solutions and data mining/machine learning research innovations for a variety of industries, such as education, telecommunications, and healthcare. He has completed nearly 8 research projects in collaboration with international universities. He has served as the journal editorial board member, track chair, session chair, and a member of the program committees of numerous conferences including. He has (co) authored 30+ publications, including 16+ journal articles, 12+ conferences, and 2 book chapters, and obtained nearly 1076+ citations with an h-index score of 14 and an i-index score of 16.

2 

Fernando Moreira was the head of the Science and Technology Department from May 2018 until February 2022. He graduated in Computer Science (1992), M.Sc. in Electronic Engineering (1997) and PhD in Electronic Engineering (2003), both at the Faculty of Engineering of the University of Porto and Habilitation (2018). He has been a member of the Science and Technology Department at Portucalense University since 1992 as a Full Professor. He teaches subjects related to undergraduate and post-graduate studies. He supervises several PhD and M.Sc. students. He is a (co-)author of more than 250 scientific publications with peer-review in national and international journals and conferences. He serves as a member of the Editorial Advisory Board for several journals and books. He organized several special issues from JCR journals. He has already regularly served on Programme and Scientific committees of national and international conferences. He was the MSc in Computation coordinator for ten years. He holds editorial experience, and he is a co-editor of several books. He is associated with NSTICC, ACM and IEEE. His principal research areas are mobile computing, ICT in Higher Education, Mobile learning, Social Business and Digital transformation. He was awarded Atlas Elsevier Award in April 2019.

3 *Received: Month DD, 20YY; Accepted: Month DD, 20YY.*