IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 3, 2021, accepted November 22, 2021, date of publication November 25, 2021,
date of current version December 7, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3130902

Subjective Answers Evaluation Using Machine Learning and Natural Language Processing

MUHAMMAD FARRUKH BASHIR[1], HAMZA ARSHAD[1], ABDUL REHMAN JAVED[2], (Member, IEEE), NATALIA KRYVINSKA[3], AND SHAHAB S. BAND[4], (Senior Member, IEEE)

[1]Faculty of Computing, Riphah International University, Islamabad 46000, Pakistan
[2]Department of Cyber Security, Air University, Islamabad 44000, Pakistan
[3]Department of Information Systems, Faculty of Management, Comenius University in Bratislava, 82005 Bratislava, Slovakia
[4]Future Technology Research Center, College of Future, National Yunlin University of Science and Technology, Yunlin 64002, Taiwan

Corresponding authors: Abdul Rehman Javed (abdulrehman.cs@au.edu.pk) and Natalia Kryvinska (natalia.kryvinska@uniba.sk)

ABSTRACT Subjective paper evaluation is a tricky and tiresome task to do by manual labor. Insufficient understanding and acceptance of data are crucial challenges while analyzing subjective papers using Artificial Intelligence (AI). Several attempts have been made to score students' answers using computer science. However, most of the work uses traditional counts or specific words to achieve this task. Furthermore, there is a lack of curated data sets as well. This paper proposes a novel approach that utilizes various machine learning, natural language processing techniques, and tools such as Wordnet, Word2vec, word mover's distance (WMD), cosine similarity, multinomial naive bayes (MNB), and term frequency-inverse document frequency (TF-IDF) to evaluate descriptive answers automatically. Solution statements and keywords are used to evaluate answers, and a machine learning model is trained to predict the grades of answers. Results show that WMD performs better than cosine similarity overall. With enough training, the machine learning model could be used as a standalone as well. Experimentation produces an accuracy of 88% without the MNB model. The error rate is further reduced by 1.3% using MNB.

INDEX TERMS Subjective answer evaluation, big data, machine learning, natural language processing, word2vec.

I. INTRODUCTION

Subjective questions and answers can assess the performance and ability of a student in an open-ended manner. The answers, naturally, are not bound to any constraint, and students are free to write them according to their mindset and understanding of the concept. With that said, several other vital differences separate subjective answers from their objective counterpart. For one, they are much longer than the objective questions. Secondly, they take more time to write. Moreover, they carry much more context and take a lot of concentration and objectivity from the teacher evaluating them.

Evaluation of such questions using computers is a tricky task, mainly because natural language is ambiguous. Several preprocessing steps must be performed, such as cleaning the data and tokenization before working on it. Then the textual data can be compared using various techniques such as document similarity, latent semantic structures, concept graphs, ontologies. The final score can be evaluated based on Similarity, keywords presence, structure, language [1], [2].

Several attempts have been made in the past to solve this problem [3]–[5], but there is still room for improvements, some of which is discussed in this paper.

Subjective exams are considered more complex and scary by both students and teachers due to their one fundamental feature, context. A subjective answer demands the checker check every word of the answer for scoring actively, and the checker's mental health, fatigue, and objectivity play a massive role in the overall result. Therefore, it is much more time and resource-efficient to let a system handle this tedious and somewhat critical task of evaluating subjective answers. Evaluating objective answers with machines is very easy and feasible. A program can be fed with questions and one-word answers that can quickly map students' responses. Nevertheless, subjective answers are much more challenging to tackle. They are varied in length and contain a vast amount of vocabulary. Furthermore, people tend to use synonyms and

The associate editor coordinating the review of this manuscript and approving it for publication was Massimo Cafaro.

convenient abbreviations, which makes the process that much tricky.

Much work has been done on the topic of subjective answers evaluation in one form or another, such as measuring Similarity between different texts, words, and even documents, finding the context behind the text and mapping it with the solution's context, counting the noun-phrase in the documents, matching keywords in the answers, and so on. However, problems such as Tf-Idf loosing semantic context [6], lack of hyper-parameters tuning [7], costly training [8], and need for better datasets [5] still exist.

In this paper, we explore a machine learning and natural language processing-based approach for subjective answers evaluation. Our work is based on natural languages processing techniques such as tokenization, lemmatization, text representing techniques such as TF-IDF, Bag of Words, word2vec, similarity measuring techniques such as cosine similarity, and word mover's distance, classification techniques such as multinomial Naive Bayes. We use different evaluation measures such as F1-score, Accuracy, and Recall to evaluate the performance of various models against each other. We also discuss various techniques used in the past for subjective answers evaluation or text similarity evaluation in general.

Following are some of the major limitations when dealing with subjective answers:

- Existing studies tend to have synonyms.
- Existing studies tend to have an extensive range of possible lengths.
- Existing studies tend to be randomly ordered among their sentences.

This paper proposes a new and improved way of evaluating descriptive question answers automatically using machine learning and natural language processing. It uses 2 step approach to solving this problem. First, the answers are evaluated using the solution and provided keywords using various Similarity-based techniques such as word mover's distance. Then the results from this step are then used to train a model that can evaluate answers without the need for solutions and keywords. For example, a subjective question ''What is the capital city of Pakistan and what is it famous for?'' can have a correct answer: ''Islamabad is the capital city of Pakistan and it is famous for mountain scenery''. Before evaluating the student's answer to the question, both the question, the answer, and also some keywords essential to the answer are fed into the system (in this case, keywords will be Islamabad and mountain scenery), and the system evaluates the student's answer by comparing both the similarity (keeping context in mind) of modal answer and student's response as well as the presence or absence of any keywords. So a student's answer of ''Karachi is the capital of Pakistan, it is famous for mountain scenery'' might get 50% marks, ''Islamabad and mountain scenery'' might get 30% marks since the main keywords are present even the context is missing and ''Islamabad is the capital and its famous for mountain scenery'' might get 100% marks since it satisfies both contextual similarities as well as keywords presence in relation to the correct answer.

A. MOTIVATION

This form of evaluation by machines is a big step forward in aiding the educational sector to perform their other duties efficiently and reduce the manual labor in trivial tasks such as comparing the answers with a correct solution. This leads to teachers spending more time teaching students, preparing a better curriculum, and evaluating their tests with less human errors and more transparency.

B. CONTRIBUTION

This paper contributes by helping solve the problem of subjective answers evaluation using machine learning and natural processing techniques. It studies various thresholds of sentence similarity measuring matrices and proposes a way to train a machine learning model, which can help reinforce confidence in evaluation scores moving forward. Other contributions include a prepared data set with solutions, answers, and keywords carefully curated by teachers.

C. PAPER ORGANIZATION

The rest of the paper is organized as follows: Section II presents the background of the problem and the literature review. Section III provides the proposed approach. Section IV presents the experimental analysis and results. Section V concludes the paper.

II. BACKGROUND AND LITERATURE REVIEW

As mentioned before, the evaluation of subjective answers is not a new thought, and it has been worked upon for almost two decades. Various techniques have been implemented to solve this problem, such as big data Natural Language Processing, Latent Semantic Analysis, Bayes theorem, K-nearest classifier, and even formal techniques such as Formal Concept Analysis. They are categorized into three main categories: Statistical, Information Extraction, and Full Natural Language Processing.

A. TECHNICAL BACKGROUND

1) STATISTICAL TECHNIQUE

It is based on keyword matching and is considered poor as it cannot tackle problems such as synonyms or take the context into account. Several works have been done on subjective paper evaluation using this approach [9], [10].

2) INFORMATION EXTRACTION (IE) TECHNIQUE

Information Extraction techniques depend on getting a structure or a pattern from the text so that the text can be broken into concepts and their relationships [11]. The dependencies found to play a significant role in producing scores and need to be confirmed from an expert in domain [12], [13].

3) FULL NATURAL LANGUAGE PROCESSING (NLP)

These techniques involve using natural language tools to parse the text and find its semantic meaning [14], [15]. That meaning can then be compared with the meaning obtained from the solution to assign the final score.

Text documents need to be processed and made ready for the machine; this step is called preprocessing and involves various natural language techniques such as Tokenization, Stopword Removal, Parts of Speech Tagging, Lemmatization, Stemming, Case Folding. Some of these techniques are briefly explained below. Madnani and Cahill [16] discussed automated scoring systems and the use of Natural Language Processing and Machine Learning in them. Lin *et al.* [17] used Natural Language Processing to measure tree similarity.

4) TOKENIZATION

Tokenization is the process of separating data into smaller parts, such as paragraphs, sentences, words, and characters. Tokenization is essential when dealing with natural language because each word must be processed separately to get its true meaning. In this work, we tokenize data into sentences and words based on white spaces and period signs. Tokenization is one of the earliest steps during natural language processing [18]. Sirts and Peekman [19] discusses evaluation results of three existing sentence segmentation and word tokenization systems on the Estonian web dataset.

5) STOPWORD REMOVAL

Natural Language has a vast vocabulary, and most elements are there for the ease of human understanding, such as 'the', 'in', 'on', 'is', and so on. These words play little to no role in most machine learning tasks and can even hinder the process by helping the model get trained on the miscellaneous data. Every language has some known stop words, which are usually removed from the corpus to make the dataset more dense and unique. Schofield *et al.* [20] argues that the use of stopword removal is superficial and that topic inference benefits little from the practice of removing stopwords beyond general terms. Çagatayli and Çelebi [21] notes that the effect of stopword removal has minimal effect on the actual results as well. However, it should be noted that frequent words with trivial importance for the machine learning model should be removed to improve the model.

6) PARTS OF SPEECH TAGGING

Parts of speech tagging is the processing of tagging each word in the data to its related part of speed, such as a noun, verb, adverb, adjective. Part of speech tagging can be done by various tools such as nltk pos tagger and helps understand the structure of the sentence. It has exciting applications such as finding noun phrases in the sentence, reducing words to their lemma, and so on. Divyapushpalakshmi and Ramar [22] used part of speech tagging for efficient sentimental analysis of Twitter.

7) LEMMATIZATION

Words found in natural language belong in many forms, such as different tense forms. For example, the words 'go', 'going', 'went' all belong to the same root word 'go' but have different forms. Lemmatization is the process of reducing all the words in the dataset to their root forms. Lemmatization requires a detailed dictionary of the words to relate them to their lemma, also called the root. It also uses part of speed information to relate the words to their specific root in the dictionary. Camastra and Razi [23] used Lemmatization and support vector machines to categorize Italian text.

8) STEMMING

Stemming is a way of reducing words to their stems, and it is based on the idea that every language has some kind of formal grammar, and the vocabulary is formed by always keeping those rules of grammar in mind. So, by using those same rules, we can reduce all the similar words back to their stems by removing their suffixes that make them different. For example, stemming plurals into singulars (words into words), stemming ending characters, and so on. There are various stemming algorithms forever in every language, such as Potter's algorithm for English word stemming. Jabbar *et al.* [24] discusses various stemming algorithms used to stem textual data.

9) CASE FOLDING

The natural language contains words in different cases, often duplicating the exact words based on their case. Therefore, it is common to reduce all the data into the same case, usually lower case, so that the machine can interpret every word in the same manner.

After the preprocessing has been done on the data according to requirements, textual data is converted in a numerical form because machines only understand numbers and understand them very well. This process is called word embedding, and some of the techniques used involve Bag of Words, TF-IDF, word2vec.

10) BAG OF WORDS (BoW)

Bag of Words is a naive technique that involves representing the vocabulary of the textual data in the form of a vector. That vector contains the index number representing either the count or the particular word at that index in the text. BoW keeps count of frequencies of words but loses the context of words. One example of BoW is a one-hot vector. Aryal *et al.* [25] used bag-of-words (BoW) vector representation to measure the similarity of two documents concerning each term occurring in the documents.

11) TERM FREQUENCY-INVERSE DOCUMENT FREQUENCY (TF-IDF)

TF-IDF is similar to BoW, where it counts the frequencies of all words present in the document, but it also keeps track of how many different sentences have those words.

This way, it provides information about the count and the value of a word in the document. Sammut *et al.* [26] discusses Tf-Idf in detail. Havrlant and Kreinovich [27] gives a probabilistic explanation of TF-IDF approach. Thakkar and Chaudhari [28] used TF-IDF to predict stock trends.

12) Word2Vec

Word2vec is a technique that uses a neural network model to learn word associations from a large dataset. It can be trained for high dimensions such as 300, which helps keep the words' semantic meaning very much intact. After the training is complete, a word2vec model can detect synonymous words or suggest other words based on the sentence. One example of a pre-trained word2vec model is Google News' 300 dimension word2vec model that contains around 100 Billion words.

After the text has been converted into numerical form, aka vectors, it is time to compare those vectors and find the Similarity of dissimilarity between them. Some of the majorly used methods for this task are Cosine Similarity, Jacquard similarity, and Word Mover's Distance. Figure 1 illustrate Word2Vec embedding. Jin *et al.* [29] studied a semantic similarity computation method based on Word2vec.

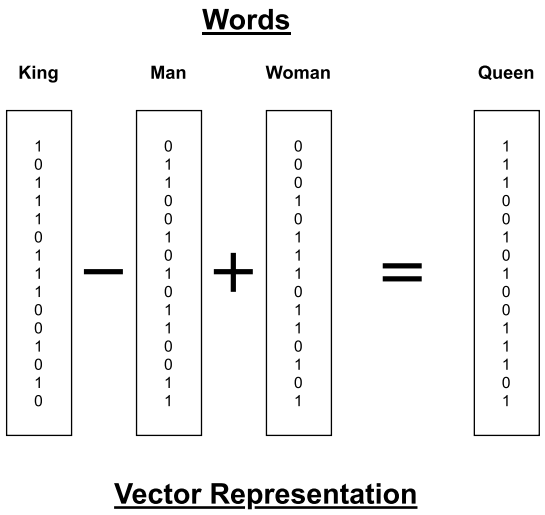

FIGURE 1. Illustration of Word2Vec embedding.

13) COSINE SIMILARITY

"Cosine similarity is a measure of Similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. A cosine angle between two vectors is measured, and its value lies between 0 and 1, 1 representing a full match. Park *et al.* [30] introduced a cosine similarity-based approach to improving the performance of conventional classifiers such as MNB, SVM, and CNN. The cosine of 0° is 1, and it is less than 1 for any other angle in the interval." this method is used extensively in the task of text processing.

14) JACQUARD SIMILARITY

The jacquard similarity is the ratio of intersection to union regarding mutual words. It finds the union of two texts and finds their intersecting terms. Then divide the intersection by its union. The higher the result, the more common words and the bigger the intersection.

15) WORD MOVER's DISTANCE (WDM)

Word Mover's Distance tries to measure the semantic distance of two documents, and word2vec embeddings bring the semantic measurement. Specifically, skip-gram word2vec is utilized in their experiments. Once the word embeddings are obtained, the semantic distance among documents is defined by the following three parts: document representation, similarity metric, and a (sparse)flow matrix. It has been shown to outperform many of the state-of-the-art methods in k-nearest neighbors classification [7]. Sato *et al.* [31] notes that WMD should be superior to BOW because WMD can take the underlying geometry into account, whereas BOW cannot.

The similarities obtained from these methods are essentially what we need in order to evaluate a subjective answer.

B. LITERATURE REVIEW

Hu and Xia [6] proposed a Latent Semantic Indexing approach for the assessment of subjective questions online. They used **Chinese** automatic segmentation techniques and subjective ontologies to make a k-dimensional LSI space matrix. The answers were presented in TF-IDF embedding matrices, and then Singular Value Decomposition (SVD) was applied to the term-document matrix, which formed a semantic space of vectors. LSI played the role of reducing problems with synonym and polysemy. At last, the Similarity between answers was calculated using cosine similarity. Dataset consisted of 35classes and 850 instances marked by teachers, and the results showed a 5% difference in grading done by teacher and the proposed system.

Kusner *et al.* [7] presented a novel concept of using Word Mover's Distance (WMD) to find the dissimilarity between two texts. The system used **no hyper-parameters** and used a relaxed WMD approach to loosen up the vector space bounds. Dataset included eight real-world sets, including Twitter sentiment data and BBC sports articles. Word2vec model from google news was used, and two other custom models were trained. KNN classification approach was used to classify the testing data. As a result, relaxed WMD reduced the error rates and led to 2 to 5 times faster classification.

Kim *et al.* [32] proposed a method to grade **short descriptive answers** lexico-semantic pattern (LSP) due to its good performance with morphologically complex Korean language. LSP can structure the semantic of the answer to help understand the user's intentions. A synonym list was also utilized to help expand the keywords so they match various

answer styles. Dataset was obtained from 88 students and converted to LSP, which was later compared with the solution LSP to score the answer. As a result, the system performed better than the existing system by 0.137

Oghbaie and Zanjireh [33] proposed a **pair-wise Similarity** measure to measure the similarity between two documents based on the keywords which appear in at least one of the documents. The work proposed a new similarity measure called PDSM (pair-wise document similarity measure), a modified version of the preferable properties approach. The proposed similarity measure was applied to text mining applications such as documents detection, k Nearest Neighbors (kNN) for single-label classification, and K-means clustering. An evaluation measure of accuracy was used, and as a result, the PDSM method produced better results than other measures like the Jaccard coefficient by 0.08 recall.

Orkphol and Yang [34] used the word2vec approach to represent words on a fix-sized vector space model and then measured the Similarity of sentences using a cosine similarity measure. Word2vec from google was used, and the sentence vector was obtained as a result of an average of words in the sentence. The score was accepted if it passed a specified threshold for similarity results, between 0 and 1. Evaluation measure of recall and accuracy was used, and as a result, the system's performance was 50.9% with and 48.7% without the probability of sense distribution.

Xia *et al.* [8] combined the word2vec approach with the **legal document** corpus to identify similarities between different law documents. Cosine similarity was used to measure the Similarity between different sentence vectors. As a result, word2vec improved the accuracy by 0.2 compared to the Bag of Words approach, which could further be increased by 0.05-0.10 by training the word2vec model on law documents.

Wagh and Anand [35] proposed a multi-criteria decision-making perspective to find the Similarity between legal documents. The work included using Artificial Intelligence and aggregation techniques such as ordered weighted average (OWA) for obtaining the similarity value between different documents. Dataset was obtained from Indian Supreme Court case judgments from years ranging from 1950 to 1993. Evaluation measures of F1score and recall were used. As a result, a concept-based similarity approach such as the one proposed in the work performed better than other techniques such as TF-IDF, getting an F1-score of up to 0.8.

Alian and Awajan [36] studied various factors affecting sentence **similarity and paraphrasing** identification using different word embedding models, clustering algorithms, and weighting methods to find the context of sentences. Pre-trained embeddings included AraVec and FastTex, both trained for the Arabic language. The Arabic training dataset included around 77,600,000 tweets. As a result, pre-trained embedding with labeled data from experts provided better recall and precision of 0.87 and 0.782 for K-means and agglomerative clustering.

Muangprathub *et al.* [5] proposed a novel approach of plagiarism detection using **formal concept analysis** (FCA). The work showed formal context in FCA, starting with two sets containing elements with some attributes that somehow relate the element to its set. The documents and their shared keywords formed a group set in FCA whose values are typically but not limited to 0 and 1. The approached used a many-valued context. The work also introduced a new similarity concept that uses both the object extent and attribute intent. The approach used is not normally utilized in similarity analysis and ranks similar documents because they have a similar object and attribute intents. The proposed system detected plagiarism in documents with 94% accuracy.

Jain and Lobiyal [37] proposed a novel approach for subjective questions evaluation using concept graphs. Concept graphs were created for both the solution and the answer, and the score was evaluated using various graph similarity techniques. Montes *et al.* [38] explained various techniques to find Similarities between concept graphs and information retrieval from such graphs.

Bahel and Thomas [39] presented an architecture for evaluation of subjective questions using text summarization, text semantics, and keywords summarization and compared the results with existing approaches. The results showed an error of 1.372 compared to 1.312 error from Jaccard's similarity approach. The approach, however, failed to compute nontextual data such as diagrams, images, and other formats.

Table 1 shows the summary of the literature review.

We studied various methods used for subjective answer evaluation in the past and looked at their shortcomings. In this paper, we propose a new approach to solve this problem which consists of training a machine learning classification model with the help of results obtained from our result prediction module and then using our trained model to reinforce results from the prediction model, which can lead to a fully trained machine learning model.

The result prediction module uses a word embedding technique called word2vec to generate vectors of our data while keeping their semantic meaning intact and then calculating the Similarity between those vectors using Word Mover's Distance. We compare our results with other methods like using Cosine Similarity and study the effect of using various machine learning models.

III. PROPOSED METHODOLOGY

The proposed system consists of data collection and annotation, preprocessing module, similarity measurement module, model training module, results predicting module, machine learning model module, and final result predicting module. First, the inputs are being taken from the user, which consists of keywords, solutions, and answers. Figure 2 shows the proposed system.

A. KEYWORDS

Keywords are question-specific things that are essential for answering that question. These keywords play a significant role in penalizing or promoting the score evaluated by the

TABLE 1. Summary of literature review.

Ref.	Evaluation Matrix	Model	Contribution	Limitations
[6]	Accuracy	LSI, SVD	LSI reduced problems with polysemy	TF-IDF loses semantic
[7]	Accuracy, Recall	WMD, KNN	Relaxed WMD better than WMD	No hyper-parameters tuning
[32]	Precision, Recall	LSP	LSP better for morphologically complexity	Need Dictionary-based LSP
[33]	Accuracy	PDSM, KNN	PDSM method than Jaccard coefficient	Domain Dependent Performance
[34]	Accuracy, Recall	Word2vec, Cosine Similarity	Fix-sized vector space model	Need better lexicon resources
[8]	Accuracy	Word2vec, Cosine Similarity	Word2vec, 0.2 better than BoW	Too much training cost
[40]	Accuracy, Recall	Jaccard & Cosine, Word2vec	Cosine Similarity better than Jaccard	Could use Programmable G-Arrays
[35]	Recall, F1-Score	AI Aggregation, OWA	Concept based similarity better than TF-IDF	Better weighting scheme
[36]	Recall, Precision	Word Embedding, Clustering	Pre-trained embedding better than K-means	Huge training time
[5]	Accuracy	FCA	FCA uses both object extent & attribute intent	New approach needs datasets

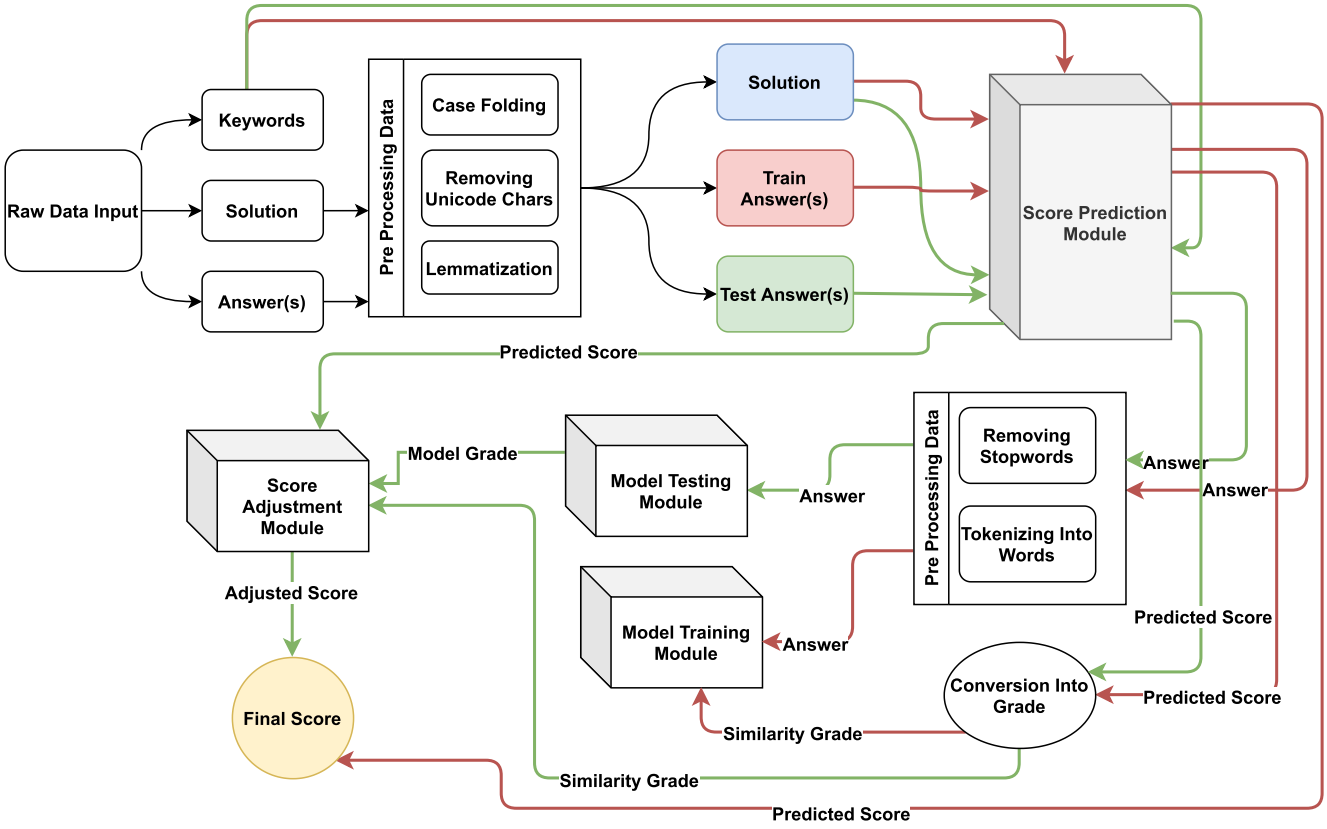

FIGURE 2. SAES proposed system.

similarity measurement module and must only contain the essential words in lower case.

B. SOLUTION

The solution is a subjective answer that is being used to map students' responses. This solution must contain all the keywords and contexts discussed in the answers in separate lines/paragraphs. The teacher/evaluator typically prepares the solution to the question.

C. ANSWER

The answer is a subjective response from the student that is to be evaluated. It usually contains some or all of the keywords and spans 1 to a few sentences depending on the type of question and the student's writing style. It almost always contains synonym words compared to the solution and, therefore, requires much more semantic care when processing.

D. DATA COLLECTION

To train and test the proposed model, there is a need for a massive amount of corpus containing subjective question answers, but there is no publicly available labeled subjective question answers corpus to the best of our knowledge. In this work, we create subjective answers labeled corpus. For generating corpus, the important thing is to target those websites and blogs where subjective questions and answers exist. We crawl various websites and collect a subjective

question answers corpus, and the crawl data belong to various domains such as computer science and general knowledge.

E. DATA ANNOTATION

After getting crawled data, there is a further need to annotate data because that crawled data is unlabeled. To annotate data, a group of different volunteers is selected, which belong to the domain of our subjective question answers corpus. We hire 30 different annotators from different colleges and universities and reside in Pakistan's different cities. Most of them are students and teachers. The average age of annotators is in the 21-25 range, whereas some annotators are in the age range of 27-51. We task annotators to best score the subjective question answers according to the answers given by students.

1) KEYWORD GENERATION

At the beginning phase of annotation, the data contains just plain answers and no specific keywords. We task annotators to identify the essential terms from the solution which can make or break the overall score of that question. These keywords help decide whether a student has mentioned relevant information in their subjective answers or not.

2) DATA ANNOTATION QUALITY VALIDATION

Data validation is crucial to obtaining accurate performance. We perform annotation from three distinct annotators of a single example. We keep the majority voted score as the final annotated label for a particular example.

3) CORPUS STATISTICS

Our annotated corpus contained over 1,000 short subjective questions, each containing a correct answer (solution) and 20 students' answers to the question, all of which were annotated. The corpus also contained necessary keywords regarding subjective questions which were extracted from the solutions.

F. PREPROCESSING MODULE

After taking inputs from the user, both the solution and the answer go through some preprocessing steps, which involve tokenization, stemming, lemmatization, stop words removal, case folding, finding, and attaching synonyms to the text. Note that stop words are not removed when passing the data to word2vec because word2vec contains a vast vocabulary and can utilize those stop words to make better semantic sense of the text. However, stop words are removed before passing to a machine learning model such as Multinomial Naive Bayes because they hinder the machine's ability to learn the patterns.

G. SIMILARITY MEASUREMENT MODULE

This module consists of WDM and Cosine Similarity functions which take two sentences or word vectors and return their Similarity. WDM tells us the dissimilarity while Cosine Similarity measures Similarity. Our approach uses both of these similarity measures one at a time and compares the results at the end. Various similarity (or dissimilarity) thresholds used are given in Table 2.

1) THRESHOLDS ANALYSIS

Various thresholds used in this paper have been experimentally deduced to produce the optimal result. WDM thresholds of WDM_LOWER and WDM_UPPER represent the dissimilarity between two sentences, where more dissimilarity represents high similarity. 0.7 threshold for WDM_LOWER was experimentally observed to represent semantically very similar sentences, and 1.6 thresholds for WDM_UPPER were observed to represent semantically less similar sentences. Anything beyond 1.6 is assumed to be too dissimilar to consider viable for comparison.

Similarly, Cosine similarity thresholds COS_LOWER and COS_UPPER represent the similarity between two sentences. It should be noted that cosine similarity does not take the context of two sentences into account when measuring similarity as opposed to WDM, hence the usage of both of these similarity (or dissimilarity) measuring approaches.

H. RESULT PREDICTING MODULE

Result Predicting Module is the core of this work. Figure 3 shows the working of this module. It operates on the following Algorithm 1:

We now have the overall score calculated by our module using either WDM or Cosine Similarity while considering the maximum matched solution/answer sentence pairs.

This result can be compared to an actual score or fed into a machine learning model to be trained.

I. MACHINE LEARNING MODEL MODULE

This model consists of Machine learning models trained on the data obtained from the result prediction module. Its working is as follows:

- Input data from Result Prediction Module.
- Preprocess the solution and answer, removing stop words, and use Countvectorizer to represent them in either Bag of Words or TF-IDF form.
- Convert the *overall score* obtained from Result Prediction Module into some category. Four categories A, B, C, and D, are used in the paper, representing 1st, 2nd, 3rd, and 4th quarter of a 100. For example, A represents marks from 0 to 25, and B represents 26 to 50.
- The number of categories is kept to a minimum because of the unavailability of the actual dataset. Practically, these categories can be extended to cover smaller score ranges.
- A machine learning model such as Multinomial Naive Bayes, which performs well for multi-class classification, is chosen.
- The preprocessed answer is used as testing data with the machine learning model to predict its class/category, and that category is checked with the result obtained from Result Prediction Module. This gives us confidence in the predicted result from the model.

TABLE 2. Similarity thresholds.

Threshold Notation	Threshold Value	Threshold Description
WDM_LOWER	0.7	Dissimilarity between two sentences using WDM is <= 0.7 meaning sentences are semantically very similar.
WDM_UPPER	1.6	Dissimilarity between two sentences using WDM is <= 1.6 meaning sentences are semantically a little bit similar.
COS_LOWER	0.2	Similarity between two sentences using Cosine Similarity is >= 0.2 meaning sentences are semantically a little bit similar.
COS_UPPER	0.7	Similarity between two sentences using Cosine Similarity is >= 0.7 meaning sentences are semantically very similar.

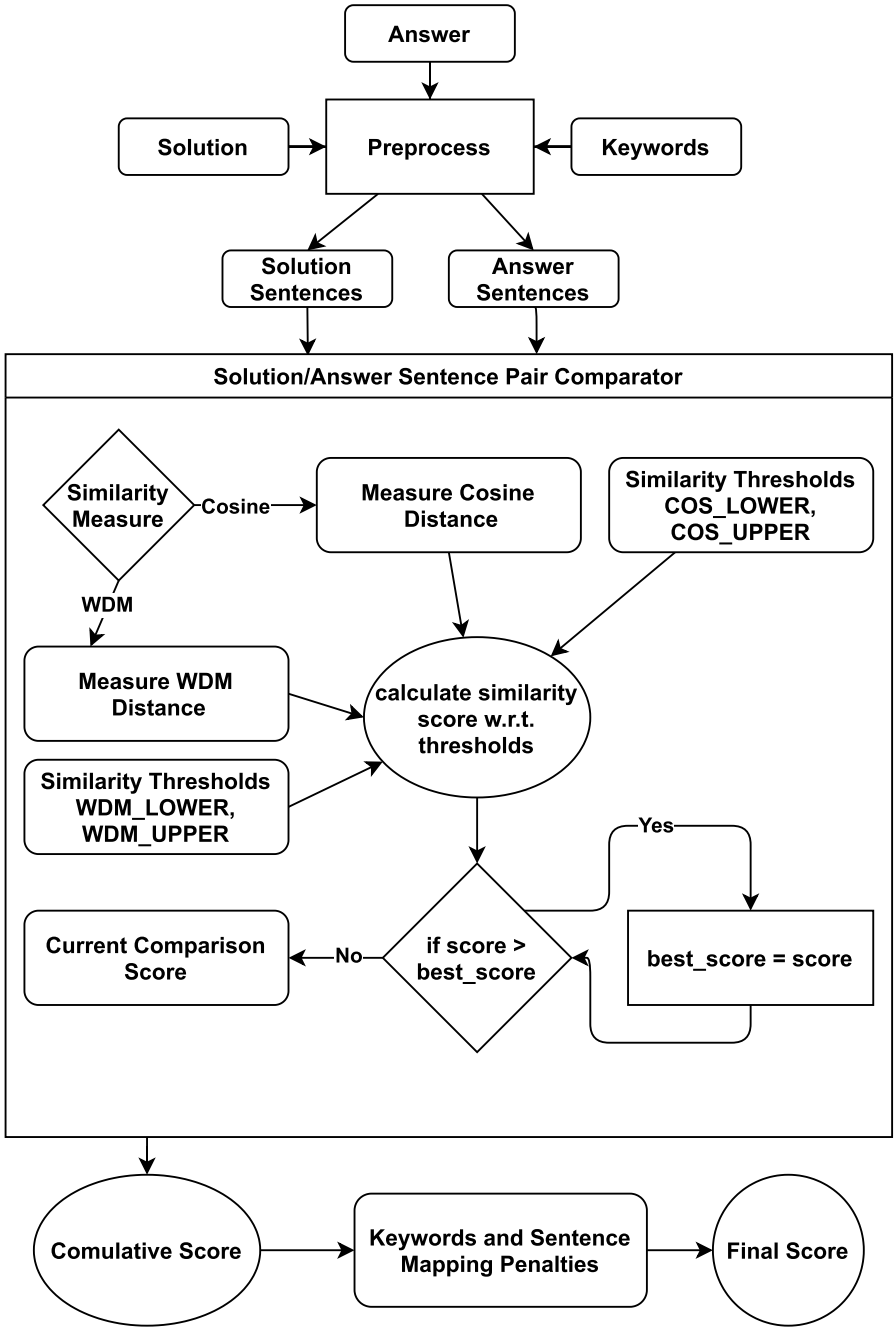

FIGURE 3. Flow chart of result prediction module.

- The preprocessed answer is fed into the machine learning model along with its label. Moreover, the model is updated according to new data.
- The predicted class is sent to the Final Score Prediction Module along with the solution, answer, and the *overall score*.

Algorithm 1 Proposed Algorithm for Subjective Answers Evaluation

We created a corpus consisting of sentences and the synonyms of sentences present in the corpus.
Sentences are composed of solution sentences and answer sentences. We defined them as: S $\in$ $S_{sen1}, A_{sen1}, S_{sen2}, A_{sen2}, \ldots, S_{senn}, A_{senn}$
We tokenize each solution sentence and answer sentence present in the corpus. Sentence and corresponding tokens are define by: Sen $\in T_1, T_2, T_3, \ldots, T_n$
After that we calculate comparison score $CS_{current}$ for every sentence S_{sen} in solution-sentences.
Compare it to every sentence A_{sen} in answer-sentences and calculate current comparison score C_{CS} in the following manner:
Keywords-weight K_w calculation:
Keep all keywords K present in S_{sen} as S_{senk}
Keep all the **K** present in both S_{sen} and A_{sen} as A_{senk}
Calculate percentage of number of keywords **K%** in A_{senk} w.r.t. S_{senk}, this shows how many **K** current A_{sen} contains w.r.t. current S_{sen}
Calculate keywords-weight K_w by dividing keywords-percentage by 100, obtaining a value between 0 and 1
Similarity distance calculation:
Calculate similarity distance S_d and similarity weight S_w between S_{sen} and A_{sen} using either one of the following methods:
Word Movers Distance W_{MD} method:
if $S_d <= W_{MD\,lower}$ **then**
    $S_w = 1 - S_d$
    $C_{CS} = S_w + K_w$
else if $S_d <= W_{MD\,upper}$ **then**
    $S_w = 1.6 - S_d$
    if $K_w >= 0.3$ (30% keywords present) **then**
        $C_{CS} = S_w + K_w$
    else if $S_w ==$ null (no keywords present in S_{sen}) **then**
        $C_{CS} = S_w$
    else if $K_w < 0.3$ (less than 30% keywords present) **then**
        $C_{CS} = 0$
    end if
end if
$CosineSimilarity C_{Sim}\, Method$:
if $S_d >= C_{Sim\,Upper}$ **then**
    $S_d = S_d$
    $C_{CS} = S_w + K_w$
else if $S_d is >= C_{Sim\,Lower}$ **then**
    if $K_w < 0.3(30\% keywords present)$ **then**
        $C_{CS} = S_w + K_w$
    else if $K_w == null$ (no keywords present in S_{sen}) **then**
        $C_{CS} = S_w$
    else if $K_w < 0.3$ (less than 30% keywords present) **then**
        $C_{CS} = 0$
    end if
end if
if $C_{CS} > CS_{current}$ **then**
    $CS_{current} = C_{CS}$
end if
Calculate overall score (0_S) by taking average of $CS_{current}$ of all S_{sen}.
Calculate missing keywords penalty (M_{Kp}) as percentage of keywords found is S_{sen} but not in A_{sen} for it's highest $CS_{current}$.
Reduce 0_S by $M_{Kp}/1.6$.
Calculate unmapped S_{sen} penalty (Um_{Ssen}) which is percentage of S_{sen} that couldn't be mapped to any A_{sen}.
Reduce O_S by Um_{Ssen}.
Calculate unmapped A_{sen} penalty Um_{Asen} which is percentage of AS that could not be mapped to any S_{sen}.
Reduce O_S by Um_{Asen}.
Return O_S as overall score of that answer.

In the beginning, the model is highly sparse and inaccurate, so it is better to train it first with a certain amount of data and then start to test it.

The advantage of the model is that it acts as a confidence booster for the Result Prediction Module, provided it has been trained on enough data. Furthermore, it can stand for its own and can be used to predict the grades/class of an answer once it has been trained on enough data. This eliminates the need for word2vec or Result Prediction Module discussed before and produced a model that can be used as a standalone evaluator for that particular question.

It also helps deal with the abnormal cases where the Result Prediction Model fails to predict the correct result for a particular answer due to semantic dissimilarity on behalf of the less trained word2vec model.

J. FINAL SCORE PREDICTION MODULE

This module is shown in Figure 4; it takes input from the machine learning module and validates the overall score with the class obtained from the machine learning module. Suppose the class matches the score. The score is considered finalized. If the class does not match the score, then the addition or deduction of half the number of values in that range is made based on whether the model suggested score is greater or lesser than the Similarity equivalent score.

It is either assumed that the machine learning model is not trained enough and the score is considered true or if the model has been extensively trained, adjusted score after the model suggestion is considered final, Accepting some inaccuracy from both the Score Prediction the Machine Learning Module.

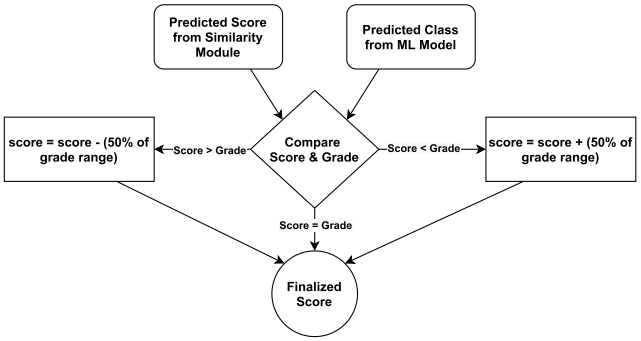

FIGURE 4. Flow chart of final score prediction module.

IV. EXPERIMENTATION AND RESULTS

The experiment setup consists of a python notebook running on a web-based Google Colab portal with a RAM of 12 GB and an HDD of 100+ GB. No GPU is turned on for this experiment.

A pre-trained word2vec model from Google consisting of 300 dimensions of around 100 Billion words vocabulary is used for this experiment. Corpus was divided into 8:2 ratio representing test and train data, respectively. Train data was used to calculate initial scores from the score prediction modules and train the machine learning model. Afterward, testing data was fed to the system one by one, updating the machine learning model.

The results are obtained using cosine similarity and word mover's distance combined with a Multinomial Naive Bayes model. Both the approaches with and without the model produced results in under a minute at Google Colab. The results are as follows.

TABLE 3. Score prediction using WDM before model suggestion.

Human Score	WDM Approach Score	Error (%)
23	33	10
74	51	23
80	52	28
20	11	9
70	83	13
10	1	9
5	0	5
0	0	0
46	32	14
60	67	7

Table 3 shows the comparison of the first ten answers used for training purposes. The score prediction module is working fairly accurately, achieving 88% accuracy. This much accuracy is significant because of word2vec in this case, and it can capture the semantic meaning of answers so well that it gives us very well Similarity among answers. Furthermore, if word2vec lacks inconsistent answers, keyword mapping and unmapped sentences thresholds still give a satisfactory score to the answers.

Table 4 shows the error when evaluating subjective answers with and without involving the model. It shows that the average errors decrease from 15.6% to 13.94% when using model suggestions for this small data set. The model's confidence level is likely to increase from 64% as the model keeps training more and more on the answers. This is a good feature of the proposed system, which leverages machine learning models to give confidence and suggestion to Similarity induced scores.

TABLE 4. Score prediction using WDM with model suggestion.

Human Score	Error Without Model	Error With Model
46	22	9.5
46	17	4.5
60	13	25.5
60	14	26.5
55	9	3.5
55	25	12.5
27	22	9.5
0	0	12.5
77	40	27.5
27	26	13.5

TABLE 5. Score prediction using cosine similarity before model suggestion.

Human Score	Cosine Approach Score	Error %age
23	33	10
74	72	2
80	72	8
20	34	14
70	95	25
10	17	7
5	0	5
0	9	9
46	34	12
60	79	19

Table 5 shows the errors in scores evaluated using the cosine similarity approach without any model suggestion. The results show an accuracy of 87%, primarily attributed to the proposed algorithm where keywords and sentence mapping play a massive role in the end. Cosine similarity performs poorly semantic-wise compared to WDM but can make some pretty good estimates where semantics are unnecessary.

TABLE 6. Score prediction using cosine similarity with model suggestion.

Human Score	Error Without Model	Error With Model
46	13	0.5
46	13	0.5
60	18	30.5
60	18	30.5
55	9	3.5
55	24	11.5
27	19	6.5
0	13	25.5
77	27	14.5
27	1	13.5

Table 6 shows the difference in errors resulting from the machine learning model correction. It shows that the model's accuracy decreased by 1.54% when using cosine similarity along with classification models. This is because the results obtained by cosine similarity are semantically weak, and the

model cannot get trained on the correct data as it does for the WDM case. Cosine similarity paired with a machine learning model yields 86% accuracy for this short dataset. Figure 5 shows the comparison of accuracy obtained from various combinations.

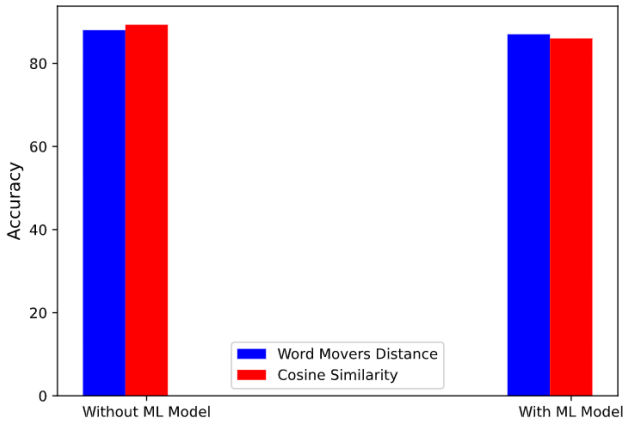

FIGURE 5. **Accuracy comparison of different models.**

V. CONCLUSION

This paper proposed a novel approach to subjective answers evaluation based on machine learning and natural language processing techniques. Two score prediction algorithms are proposed, which produce up to 88% accurate scores. Various similarity and dissimilarity thresholds are studied, and various other measures such as the keyword's presence and percentage mapping of sentences are utilized to overcome the abnormal cases of semantically loose answers. The experimentation results show that on average word2vec approach performs better than traditional word embedding techniques as it keeps the semantics intact. Furthermore, Word Mover's Distance performs better than Cosine Similarity in most cases and helps train the machine learning model faster. With enough training, the model can stand on its own and predict scores without the need for any semantics checking.

In terms of future improvements, the word2vec model can be trained especially for subjective answers evaluation of a particular domain, and with large data sets, the number of classes or grades in the model can be significantly increased. Subjective answers evaluation remains an interesting problem to tackle, and in the future, we hope to find more efficient ways to solve this problem.

REFERENCES

[1] J. Wang and Y. Dong, "Measurement of text similarity: A survey," *Information*, vol. 11, no. 9, p. 421, Aug. 2020.
[2] M. Han, X. Zhang, X. Yuan, J. Jiang, W. Yun, and C. Gao, "A survey on the techniques, applications, and performance of short text semantic similarity," *Concurrency Comput., Pract. Exper.*, vol. 33, no. 5, Mar. 2021.
[3] M. S. M. Patil and M. S. Patil, "Evaluating Student descriptive answers using natural language processing," *Int. J. Eng. Res. Technol.*, vol. 3, no. 3, pp. 1716–1718, 2014.
[4] P. Patil, S. Patil, V. Miniyar, and A. Bandal, "Subjective answer evaluation using machine learning," *Int. J. Pure Appl. Math.*, vol. 118, no. 24, pp. 1–13, 2018.
[5] J. Muangprathub, S. Kajornkasirat, and A. Wanichsombat, "Document plagiarism detection using a new concept similarity in formal concept analysis," *J. Appl. Math.*, vol. 2021, pp. 1–10, Mar. 2021.
[6] X. Hu and H. Xia, "Automated assessment system for subjective questions based on LSI," in *Proc. 3rd Int. Symp. Intell. Inf. Technol. Secur. Informat.*, Apr. 2010, pp. 250–254.
[7] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger, "From word embeddings to document distances," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 957–966.
[8] C. Xia, T. He, W. Li, Z. Qin, and Z. Zou, "Similarity analysis of law documents based on Word2vec," in *Proc. IEEE 19th Int. Conf. Softw. Qual., Rel. Secur. Companion (QRS-C)*, Jul. 2019, pp. 354–357.
[9] H. Mittal and M. S. Devi, "Subjective evaluation: A comparison of several statistical techniques," *Appl. Artif. Intell.*, vol. 32, no. 1, pp. 85–95, Jan. 2018.
[10] L. A. Cutrone and M. Chang, "Automarking: Automatic assessment of open questions," in *Proc. 10th IEEE Int. Conf. Adv. Learn. Technol.*, Sousse, Tunisia, Jul. 2010, pp. 143–147.
[11] G. Srivastava, P. K. R. Maddikunta, and T. R. Gadekallu, "A two-stage text feature selection algorithm for improving text classification," Tech. Rep., 2021.
[12] H. Mangassarian and H. Artail, "A general framework for subjective information extraction from unstructured English text," *Data Knowl. Eng.*, vol. 62, no. 2, pp. 352–367, Aug. 2007.
[13] B. Oral, E. Emekligil, S. Arslan, and G. Eryiğit, "Information extraction from text intensive and visually rich banking documents," *Inf. Process. Manage.*, vol. 57, no. 6, Nov. 2020, Art. no. 102361.
[14] H. Khan, M. U. Asghar, M. Z. Asghar, G. Srivastava, P. K. R. Maddikunta, and T. R. Gadekallu, "Fake review classification using supervised machine learning," in *Proc. Pattern Recognit. Int. Workshops Challenges (ICPR)*. New York, NY, USA: Springer, 2021, pp. 269–288.
[15] S. Afzal, M. Asim, A. R. Javed, M. O. Beg, and T. Baker, "URLdeepDetect: A deep learning approach for detecting malicious URLs using semantic vector models," *J. Netw. Syst. Manage.*, vol. 29, no. 3, pp. 1–27, Mar. 2021.
[16] N. Madnani and A. Cahill, "Automated scoring: Beyond natural language processing," in *Proc. 27th Int. Conf. Comput. Linguistics (COLING)*, E. M. Bender, L. Derczynski, and P. Isabelle, Eds. Santa Fe, NM, USA: Association for Computational Linguistics, Aug. 2018, pp. 1099–1109.
[17] Z. Lin, H. Wang, and S. I. McClean, "Measuring tree similarity for natural language processing based information retrieval," in *Proc. Int. Conf. Appl. Natural Lang. Inf. Syst. (NLDB)* (Lecture Notes in Computer Science), vol. 6177, C. J. Hopfe, Y. Rezgui, E. Métais, A. D. Preece, and H. Li, Eds. Cardiff, U.K.: Springer, 2010, pp. 13–23.
[18] G. Grefenstette, "Tokenization," in *Syntactic Wordclass Tagging*. Springer, 1999, pp. 117–133.
[19] K. Sirts and K. Peekman, "Evaluating sentence segmentation and word Tokenization systems on Estonian web texts," in *Proc. 9th Int. Conf. Baltic (HLT)* (Frontiers in Artificial Intelligence and Applications) vol. 328, U. Andrius, V. Jurgita, K. Jolantai, and K. Danguole, Eds. Kaunas, Lithuania: IOS Press, Sep. 2020, pp. 174–181.
[20] A. Schofield, M. Magnusson, and D. M. Mimno, "Pulling out the stops: Rethinking stopword removal for topic models," in *Proc. 15th Conf. Eur. Chapter Assoc. Comput. Linguistics (EACL)* vol. 2, M. Lapata, P. Blunsom, and A. Koller, Eds. Valencia, Spain: Association for Computational Linguistics, 2017, pp. 432–436.
[21] M. Çagatayli and E. Çelebi, "The effect of stemming and stop-word-removal on automatic text classification in Turkish language," in *Proc. 22nd Int. Conf. Neural Inf. Process. (ICONIP)* (Lecture Notes in Computer Science), vol. 9489, S. Arik, T. Huang, W. K. Lai, and Q. Liu, Eds. Istanbul, Turkey: Springer, 2015, pp. 168–176.
[22] M. Divyapushpalakshmi and R. Ramalakshmi, "An efficient sentimental analysis using hybrid deep learning and optimization technique for Twitter using parts of speech (POS) tagging," *Int. J. Speech Technol.*, vol. 24, no. 2, pp. 329–339, Jun. 2021.
[23] F. Camastra and G. Razi, "Italian text categorization with lemmatization and support vector machines," in *Neural Approaches to Dynamics of Signal Exchanges* (Smart Innovation, Systems and Technologies), vol. 151, A. Esposito, M. Faúndez-Zanuy, F. C. Morabito, and E. Pasero, Eds. New York, NY, USA: Springer, 2020, pp. 47–54.
[24] A. Jabbar, S. Iqbal, M. I. Tamimy, S. Hussain, and A. Akhunzada, "Empirical evaluation and study of text stemming algorithms," *Artif. Intell. Rev.*, vol. 53, no. 8, pp. 5559–5588, Dec. 2020.

[25] S. Aryal, K. M. Ting, T. Washio, and G. Haffari, "A new simple and effective measure for bag-of-word inter-document similarity measurement," 2019, *arXiv:1902.03402*.
[26] C. Sammut and G. I. Webb, "TF-IDF," in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds. Springer, 2010, pp. 986–987.
[27] L. Havrlant and V. Kreinovich, "A simple probabilistic explanation of term frequency-inverse document frequency (tf-idf) heuristic (and variations motivated by this explanation)," *Int. J. Gen. Syst.*, vol. 46, no. 1, pp. 27–36, Mar. 2017.
[28] A. Thakkar and K. Chaudhari, "Predicting stock trend using an integrated term frequency–inverse document frequency-based feature weight matrix with neural networks," *Appl. Soft Comput.*, vol. 96, Nov. 2020, Art. no. 106684.
[29] X. Jin, S. Zhang, and J. Liu, "Word semantic similarity calculation based on Word2vec," in *Proc. Int. Conf. Control, Autom. Inf. Sci. (ICCAIS)*, Hangzhou, China, Oct. 2018, pp. 12–16.
[30] K. Park, J. S. Hong, and W. Kim, "A methodology combining cosine similarity with classifier for text classification," *Appl. Artif. Intell.*, vol. 34, no. 5, pp. 396–411, Apr. 2020.
[31] R. Sato, M. Yamada, and H. Kashima, "Re-evaluating word mover's distance," 2021, *arXiv:2105.14403*.
[32] J. E. Kim, K. Park, J. M. Chae, H. J. Jang, B. W. Kim, and S. Y. Jung, "Automatic scoring system for short descriptive answer written in Korean using Lexico-semantic pattern," *Soft Comput.*, vol. 22, no. 13, pp. 4241–4249, 2018.
[33] M. Oghbaie and M. M. Zanjireh, "Pairwise document similarity measure based on present term set," *J. Big Data*, vol. 5, no. 1, pp. 1–23, Dec. 2018.
[34] K. Orkphol and W. Yang, "Word sense disambiguation using cosine similarity collaborates with Word2vec and WordNet," *Future Internet*, vol. 11, no. 5, p. 114, May 2019.
[35] R. S. Wagh and D. Anand, "Legal document similarity: A multicriteria decision-making perspective," *PeerJ Comput. Sci.*, vol. 6, p. e262, Mar. 2020.
[36] M. Alian and A. Awajan, "Factors affecting sentence similarity and paraphrasing identification," *Int. J. Speech Technol.*, vol. 23, no. 4, pp. 851–859, Dec. 2020.
[37] G. Jain and D. K. Lobiyal, "Conceptual graphs based approach for subjective answers evaluation," *Int. J. Conceptual Struct. Smart Appl.*, vol. 5, no. 2, pp. 1–21, Jul. 2017.
[38] M. Montes-y-Gómez, A. López-López, and A. Gelbukh, "Information retrieval with conceptual graph matching," *Proc. Int. Conf. Database Expert Syst. Appl.*, vol. 1873, Jan. 2000, pp. 312–321.
[39] V. Bahel and A. Thomas, "Text similarity analysis for evaluation of descriptive answers," 2021, *arXiv:2105.02935*.
[40] A. W. Qurashi, V. Holmes, and A. P. Johnson, "Document processing: Methods for semantic text similarity analysis," in *Proc. Int. Conf. Innov. Intell. Syst. Appl. (INISTA)*, Aug. 2020, pp. 1–6.

MUHAMMAD FARRUKH BASHIR received the master's degree in computer science from the National University of Computer and Emerging Sciences, Islamabad, Pakistan. He worked as a Visiting Lecturer at the Department of Computer Science, Air University, Islamabad. He is currently a Lecturer with Riphah International University, Islamabad. His research interests include data science, data mining and machine learning, big data management, natural language processing, computer vision, and explainable AI.

HAMZA ARSHAD is currently pursuing the B.S. degree in software engineering with Abasyn University, Islamabad. He is working on his final year project. His current research interests include machine learning, natural language processing, and computer vision. He has a keen interest in software products and aims to produce simple, clean, and robust applications for daily life problems.

ABDUL REHMAN JAVED (Member, IEEE) received the master's degree in computer science from the National University of Computer and Emerging Sciences, Islamabad, Pakistan. He has worked with the National Cybercrimes and Forensics Laboratory, Air University, Islamabad. He is a cybersecurity researcher and a practitioner with industry and academic experience. He is currently a Lecturer with the Department of Cyber Security, Air University. He is supervising/co-supervising several graduate (B.S. and M.S.) students on topics related to health informatics, cybersecurity, mobile computing, and digital forensics. He has authored over 50 peer-reviewed research articles. His current research interests include but are not limited to mobile and ubiquitous computing, data analysis, knowledge discovery, data mining, natural language processing, smart homes, and their applications in human activity analysis, human motion analysis, and e-health. He aims to contribute to interdisciplinary research of computer science and human-related disciplines. He is a member of ACM. He is a TPC Member of CID2021 (Fourth International Workshop on Cybercrime Investigation and Digital forensics—CID2021) and the 44th International Conference on Telecommunications and Signal Processing. He has served as a Moderator for the 1st IEEE International Conference on Cyber Warfare and Security (ICCWS). He has reviewed over 150 scientific research articles for various well-known journals, including IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, *Computers and Electrical Engineering* (Elsevier), *Sustainable Cities and Society* (Elsevier), *Journal of Information Security and Applications* (Elsevier), *IEEE Internet of Things Magazine*, IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING, *ACM Transactions on Internet Technology*, *Telecommunication Systems* (Springer), IEEE ACCESS, and *International Journal of Ad Hoc and Ubiquitous Computing* (Inderscience).

NATALIA KRYVINSKA received the Ph.D. degree in electrical and IT engineering from the Vienna University of Technology, Austria. She also received a Docent title (Habilitation) in management information systems from Comenius University in Bratislava, Slovakia, and a Professor title and was appointed for the professorship by the President of the Slovak Republic. She is currently a Full Professor and the Head of the Department of Information Systems, Faculty of Management, Comenius University in Bratislava, Slovakia. Previously, she served as a University Lecturer and a Senior Researcher for the eBusiness Department, School of Business Economics and Statistics, University of Vienna. Her research interests include complex service systems engineering, service analytics, and applied mathematics.

SHAHAB S. BAND (Senior Member, IEEE) received the M.Sc. degree in artificial intelligence from Iran, and the Ph.D. degree in computer science from the University of Malaya (UM), Malaysia, in 2014. He was an Adjunct Assistant Professor with the Department of Computer Science, Iran University of Science and Technology. He also served as a Senior Lecturer for UM and Islamic Azad University, Iran. He participated in many research programs within the Center of Big Data Analysis, IUST, and IAU. He has been associated with young researchers and the elite club, since 2009. He supervised or co-supervised undergraduate and postgraduate students (master's and Ph.D.) by research and training. He has also authored or coauthored papers published in IF journals and attended high-rank A and B conferences. He is a Professional Member of the ACM. He is an associate editor, a guest editor, and a reviewer of high-quality journals and conferences.